Universidad de las Ciencias Informáticas

FACULTAD 4

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

DESARROLLO DEL VIDEO-JUEGO "RECICLA CONMIGO" DE LA COLECCIÓN MUNDOCLICK PARA DISPOSITIVOS MÓVILES CON SISTEMA OPERATIVO ANDROID.

AUTOR:

GUILLERMO GONZÁLEZ JIMÉNEZ.

TUTORES

MSC. LICET GUTIERREZ MOMPIÉ

ING. MICHAEL HORTA FLEITAS

LA HABANA, 2016

"AÑO 58 DE LA REVOLUCIÓN"


"Seamos realistas y hagamos lo Imposible."
Ernesto Che Guevara.

DECLARACIÓN DE AUTORÍA.

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas (UCI) los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente: a los ___ días del mes de _____ del año 2015.

_____

Guillermo González Jiménez

Autor

_____ _____

MSc. Licet Gutierrez Mompié **Ing. Michael Horta Fleitas**

Tutor **Tutor**

AGRADECIMIENTOS

A mi madre por su apoyo.

A mi padre por ser mi guía y ejemplo.

A mi tía Laura por ser como una madre para mí.

A mi hermano.

A mi abuela Caridad y mi abuelo Gaspar por su ejemplo incondicional.

A mi tío Rafael por estar siempre dispuesto a ayudarme.

A mi queridísima novia por su apoyo y por ser como es.

A mis amigos y a todos aquellos que de una forma u otra hicieron posible la conformación de este trabajo de diploma.

DEDICATORIA

A mi familia.

RESUMEN

El cuidado de la naturaleza en la actualidad supone una de las metas más complejas y necesarias para preservar el planeta tierra. Por este motivo se persigue instruir y motivar a los niños desde edades tempranas en la necesidad del reciclaje y recolección de los desechos. Con vistas a fortalecer el apoyo en este sentido, se desarrolló el video-juego “Recicla conmigo” de la colección MundoClick para los sistemas operativos (SO) Windows y Linux, por el centro FORTES de la facultad 4 en la Universidad de las Ciencias Informáticas. Con el objetivo de aumentar la compatibilidad de dicho juego, se estudió la forma de llevarlo al SO Android para que estuviera disponible en dispositivos móviles. Para el desarrollo del mismo se investigaron las posibles herramientas a usar, así como la metodología y juegos semejantes. Una vez concluida la investigación se logró el correcto desarrollo del video-juego, para finalmente y con vistas a validar dicha propuesta de solución, realizar pruebas de calidad al software que permitieran constatar la eficiencia del mismo.

Palabras claves: video-juego, Android, dispositivos móviles, celular, software educativo.

TABLA DE CONTENIDOS.

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Los video-juegos en la actualidad son de los productos informáticos de mayor auge en el desarrollo de software, dada la gran cantidad de usuarios que de una forma u otra los utilizan, ya sea para el entretenimiento o el aprendizaje, entre otros. Los mismos surgen en la década del 40 junto a las primeras computadoras, desarrolladas a finales de la II Guerra Mundial. En lo adelante se crean algunas compañías como ATARI, KONAMI, SEGA, entre otras, que fueron las responsables de dar fama global a estos medios de entretenimiento.

El desarrollo que han supuesto las tecnologías en la actualidad, ha traído consigo la paralela evolución de los video-juegos. Esos antiguos y entretenidos video-juegos en blanco y negro, llegan hoy en día con disímiles propuestas que en algunos casos superan los límites de la imaginación, creando inmensos mundos ficticios, con gráficos de increíble calidad. Es posible además, encontrar una inmensa gama de dispositivos en los cuales poder interactuar con estos recursos, que van desde los teléfonos móviles hasta las computadoras personales.

Con los video-juegos se puede lograr que las personas asimilen los conocimientos de forma interactiva y no los rechacen, como suele ocurrir con los métodos tradicionales de enseñanza. Está demostrado además que estos productos informáticos son capaces de mejorar las habilidades de razonamiento e incentivar a sus consumidores.

La Universidad de las Ciencias Informáticas (UCI) tiene dentro de sus objetivos como institución de la educación superior, la preparación de profesionales integrales y es por ello que la mayoría de los estudiantes y profesores se ven vinculados a los procesos de producción de software en ella. Así es como en la facultad 4 de la UCI, se crea el Centro de Tecnologías para la Formación (FORTES), que tiene como misión, el desarrollo de tecnologías y servicios que permitan la implementación de soluciones para la formación, en todo tipo de instituciones con diferentes modelos de formación y recursos tecnológicos.

Dentro de los proyectos que se desarrollan en el centro FORTES, se destaca la colección **MundoClick,** que solo se encuentra disponible para los sistemas operativos Windows y Linux. Este software es un compendio de video-juegos dirigidos a la enseñanza primaria, por medio del cual se persigue incentivar a los niños desde edades tempranas (entre 6 y 12 años de edad) en el cuidado y respeto a la naturaleza, y reforzar sus habilidades en las diferentes ramas de las ciencias: como las matemáticas, la computación, entre otras. Dentro de los juegos que conforman la colección, está el video-juego “Recicla conmigo”, el mismo persigue inculcar la necesidad del reciclaje de los desechos que comúnmente son arrojados, apoyando así a la no contaminación ambiental y a la reutilización de los recursos.

Con vistas a lograr mayor uso por parte de niños y adolescentes, el centro se dispuso a llevar el video-juego “Recicla conmigo” a dispositivos móviles con Sistema Operativo (SO) Android, por ser una de las tecnologías con mayor impacto en la actualidad.

Para el desarrollo del video-juego en SO Android se estudió la posibilidad de, por medio de herramientas existentes exportar el video-juego actual. En este sentido se pudo constatar la imposibilidad de su éxito, dado que la plataforma donde se encuentra el video-juego está desarrollada bajo el marco de trabajo **nodeJS**, que es una herramienta que requiere de un servidor web para su funcionamiento, impidiendo de esta forma el uso de softwares para exportar, que permitan convertir aplicaciones web en aplicaciones para dispositivos con SO Android. Dicha imposibilidad de exportar el proyecto no fue el único obstáculo, aun si fuera posible exportarlo, se presentarían los siguientes problemas:

1. No adaptabilidad de la resolución de la aplicación Recicla a los dispositivos móviles, dado que el software está diseñado para computadoras personales (PC).
2. No se puede hacer uso pleno de las facilidades que brindan las pantallas táctiles de los dispositivos móviles.
3. No es posible hacer uso de todos los botones internos y externos con que cuentan los dispositivos, que permiten mejorar la forma de jugar de los usuarios.
4. Imposibilidad de hacer uso de la propiedad de rotado de las aplicaciones móviles.

Por lo antes expuesto se plantea el siguiente **problema de la investigación**: ¿Cómo lograr la compatibilidad del video-juego “Recicla conmigo” de la colección MundoClick en dispositivos móviles con SO Android?

Definiéndose como **objetivo general:** Desarrollar el video-juego “Recicla conmigo” de la colección MundoClick en SO Android.

Que supone como **objeto de estudio:** el desarrollo de video-juegos, enmarcando la investigación al siguiente **campo de acción:** desarrollo de video-juegos para SO Android.

Es así que para dar cumplimiento al objetivo general se definen los siguientes **objetivos específicos:**

- Analizar las tendencias y tecnologías utilizadas en el desarrollo de video-juegos semejantes al video-juego “Recicla conmigo”.
- Analizar las herramientas utilizadas en el desarrollo de aplicaciones móviles.
- Hacer el análisis y diseño del proyecto a desarrollarse.

- Implementación del video-juego “Recicla conmigo” a partir de las herramientas definidas en el estudio previo.
- Evaluar el correcto funcionamiento del video-juego por medio de pruebas al software.

Para guiar la investigación en el presente trabajo, se formularon las siguientes **preguntas científicas**:

- ¿Cuáles son las tendencias actuales en el desarrollo de video-juegos de reciclaje?
- ¿Qué herramientas y tecnologías utilizar para el desarrollo del video-juego “Recicla conmigo” en SO Android?
- ¿Cómo realizar el análisis y diseño del video-juego?
- ¿Qué técnicas, estilos y patrones utilizar en la implementación del video-juego?
- ¿Cómo validar la propuesta de solución?

Durante la investigación se utilizaron los **métodos científicos** siguientes:

Métodos teóricos:

- Histórico-lógico: Para el estudio de los precedentes juegos educativos y las tendencias actuales de los mismo en SO Android.
- Analítico-sintético: Es empleado al hacer el análisis y estudio de los conceptos y definiciones relacionados a la investigación.
- Modelación: Utilizado al representar los procesos que se desean automatizar, así como la elaboración de los diferentes modelos definidos por la metodología.

Método empírico:

- Observación: En el análisis del video-juego “Recicla conmigo” (con que cuenta actualmente la colección MundoClick) y de los video-juegos homólogos.

Resumen de los capítulos:

Capítulo 1:

En este capítulo se hace un estudio de los video-juegos educativos, para luego llevar a cabo un análisis de los juego semejantes al video-juego “Recicla conmigo” y finalmente definir las herramientas a utilizar en el proceso de desarrollo.

Capítulo 2:

Se realiza el análisis y diseño del video-juego por medio del documento de diseño, donde se registran todas las características más importantes que tendrá el video-juego, antes de empezar el desarrollo. Además, en este capítulo se definen las iteraciones realizadas durante el desarrollo de la aplicación.

Capítulo 3:

Se documenta el proceso de desarrollo del video-juego, así como el registro de las pruebas realizadas a la aplicación, midiendo características de funcionamiento y rendimiento del software.

CAPÍTULO 1: MARCO TEÓRICO

A continuación se persigue hacer un estudio previo al desarrollo del video-juego, por medio de la investigación de temas que permitan sentar las bases del trabajo, como es el caso de las soluciones homólogas. Por ello se hace necesario hacer un estudio sobre tendencias en el campo del reciclaje en aplicaciones para Android, con vistas a validar la inexistencia de un producto similar al deseado. Una vez terminada la primera fase de la investigación, fue posible estudiar la metodología y herramientas que facilitarían el proceso de desarrollo.

1.1 Historia de los video-juegos

Sobre la década del 40 a finales de la segunda guerra mundial, se empiezan a concretar los primeros intentos del hombre por crear máquinas automáticas para el procesamiento de información. Esto se debió en gran medida, al uso de la máquina electromecánica **Bombe** diseñada por Alan Turing para descifrar los códigos nazis. Así se evidenció la importancia de automatizar los procesos por medio de máquinas, y surge entonces el término computación. Las primeras computadoras eran aparatos inmensos, que podían pesar varias toneladas y abarcar grandes espacio; ejemplo de esto es la “ENIAC”, desarrollada en la universidad de Pennsylvania, que pesaba varias toneladas y ocupaba todo un sótano. Con el paso del tiempo las computadoras fueron disminuyendo en tamaño y aumentando su procesamiento con la aparición de los primeros microchips (2; 3).

Junto a las computadoras, los video-juegos fueron madurando paralelamente, y aunque en sus inicios no fueron considerados video-juegos como tal, sino que se definían como juegos-electrónicos, puesto que su funcionamiento se basaba en un artefacto electro-mecánico (4).

Los primeros intentos de juegos-electrónicos fueron estudios e investigaciones, dado que en el momento no se veían como algo prometedor. Ejemplo de esto, fue el **OXO** desarrollado por Alexander Shafto Douglas en 1952 como complemento de su tesis de doctorado, que tenía como tema principal la interactividad entre seres humanos y computadoras. El juego consistía en el famoso “cruz y raya” y permitía jugar a un humano contra la computadora. Un poco más adelante se puede ver el juego **Tenis for two**, creado por William Higinbotham en 1958 del Laboratorio Nacional de Brookhaven el cual consistía en un juego de tenis simulado en la pantalla de un osciloscopio. Este juego tenía como objetivo entretener a los visitantes del laboratorio y por este motivo años más tarde es desmantelado para emplear sus piezas en investigaciones más serias (4).

Así es como se sentaron las bases de los juegos-electrónicos y se abren las puertas a un nuevo mundo que es el de los video-juegos, que no es hasta la década del 70 cuando toman un gran auge con la aparición de compañías dedicadas a comercializar todo tipo de artefactos para el entretenimiento. Algunas de las compañías más conocidas que aún subsisten son (5):

- **NINTENDO**: Es una empresa japonesa que en los inicios de los 70 presentó dispositivos de juegos para salas recreativas y más adelante crea una consola que permitía jugar varios video-juegos con solo conectarse a un televisor.
- **ATARI**: Es una empresa estadounidense famosa por la creación de la videoconsola doméstica Atari Pong, que se conectaba a un aparato de televisión y permitía jugar el famoso video-juego Pong.
- **KONAMI**: Empresa japonesa que en sus inicios se dedicaba a la creación de máquinas arcade y videoconsolas para otras compañías, luego se dedicó a realizar sus propios video-juegos y es una de las empresas más conocidas hasta la actualidad.

Durante algún tiempo luego de los 70, los video-juegos fueron pasando de ser jugados en consolas personalizadas que necesitaban de medios como un televisor, hacia dispositivos únicamente destinados al juego con todos los periféricos necesarios integrados. Es aquí cuando llega la década de los 90 donde las computadoras personales tienen un inmenso auge, y revolucionan la industria de los video-juegos convirtiéndola en el monopolio que representa en la actualidad. Con los juegos para computadoras los usuarios tenían nuevas funcionalidades, nuevos modos y formas que permitieron un desarrollo paulatino de los mismos (6).

Es así que con el surgimiento de nuevas tecnologías, tales como las tecnologías móviles, no fue mucha la espera para encontrarse un mercado abarrotado de video-juegos y recursos disponibles para estos medios. Ya sea para sistemas operativos iOS (de la reconocida firma Apple) como de Android (que en la actualidad se encuentra bajo la tutela de Google) o de otros no tan conocidos pero no menos importantes como Windows Mobile. Resulta que en la actualidad es posible encontrar en cualquier medio informático uno de esos apasionantes y entretenidos video-juegos que remontan su historia a los rústicos juegos de los 70.

1.2 Juegos de reciclaje

Dentro de los video-juegos desarrollados para SO Android que tienen como tema principal el reciclaje, se pueden encontrar los que a continuación se exponen en la investigación, con el objetivo de estudiar las tendencias en este campo.

1.2.1 Video-juego "*Recicle*"

Es un video-juego diseñado para el entretenimiento de niños que persigue enseñar la importancia de poner los desechos en el lugar correcto. El escenario principal del video-juego, está representado por un puente que se encuentra encima de un río, por donde las personas pasan y de forma indiscriminada arrojan desechos hacia el agua. El objetivo del usuario es interceptar los desechos antes de que toquen el agua y depositarlos en sus respectivos contenedores. A medida que van pasando los niveles se va aumentando en dificultad y es cada vez más difícil depositar los desechos en sus recipientes dado que van cayendo más rápido (7).

Como características positivas del video-juego se pueden señalar: que está disponible de forma gratuita y cuenta con diversas formas de instruir a los niños en la necesidad del reciclaje. En cuanto a los aspectos negativos, carece de historia central para atraer a los usuarios y con relación al diseño presenta una baja elaboración y acabado de las texturas e imágenes presentes. Dichas características son muy importantes a la hora de realizar un producto educativo dirigido a los niños (7).

Imagen 1: Video-juego Recicle

1.2.2 Video-juego "*Fun to recycle*"

Video-juego sencillo que consta solo de 4 vistas:

- **Menú**: vista principal de la aplicación.
- **Ventana de juego**: Aquí se presentan los 3 tipos de envases para clasificar los desechos (papel, plástico y vidrio) y encima se van mostrando los diferentes residuos que se deben identificar.
- **Ventana de ayuda**: explica brevemente las instrucciones del juego.

- **Ventana de puntuación**: se muestran las 10 mejores puntuaciones del juego, además del nombre de los usuarios que las obtienen.

Durante el juego las reglas son sencillas: por cada residuo correctamente clasificado se ganan 100 puntos, mientras que por cada residuo incorrecto se pierden 50 puntos. Además de esto, el juego es por tiempo, y durante 5 minutos hay que tratar de identificar la mayor cantidad de residuos (7).

Dentro de las características positivas que se pueden encontrar en el juego, destaca la tabla de puntuaciones con que cuenta, este recurso generalmente incita al niño a tratar de superarse cada vez más, haciéndolo inconscientemente mejorar sus habilidades para catalogar los diferentes tipos de residuos. Pese a esto las características negativas son las que más abundan en el juego, como el hecho de carecer de historia o guion, cuenta con un diseño poco elaborado y solo se encuentra en idioma inglés, lo que impide su uso por parte de los usuarios no angloparlantes (7).

Imagen 2: Video-juego "Fun to recycle"

1.2.3 Video-juego "*Taree recycles*"

Software desarrollado por **IMPACT apps**, una comunidad Australiana sin ánimos de lucro que tiene como principal objetivo el cuidado del medio ambiente. El juego se encuentra actualmente en su versión 1.0.0 y fue lanzado en el año 2016. El video-juego muestra una interesante propuesta para la clasificación de los residuos, ya que se basa en una cinta transportadora que va moviendo productos aleatorios y el usuario debe interceptarlos antes del final de la cinta y depositarlos en sus respectivos contenedores (7). Los contenedores son de 4 tipos diferentes:

1. Orgánicos.
2. Materiales reciclables.
3. Desechos del jardín.
4. Otros productos.

Este video-juego presenta varios aspectos positivos, como es el hecho de contar con un agradable diseño acorde al tema que trata, permite almacenar las mejores puntuaciones y de esta forma incentivar a los usuarios a seguir jugando y superarse (7). Como aspecto negativo, el juego solo está disponible en idioma inglés y carece de historia central.

Imagen 3: Video-juego "Taree recycles"

Resumen de los video-juegos estudiados

A partir de la investigación realizada sobre video-juegos desarrollados en Android relacionados con el reciclaje, se pueden definir los siguientes aspectos positivos para tenerse en cuenta en la realización del video-juego "Recicla conmigo":

- Orientación horizontal del video-juego (landscape) para aumentar el espacio.
- Uso de métodos nativos de los dispositivos, como el deslizamiento de las pantallas (swipe).
- Guardado de las configuración del video-juego.

Características negativas de los video-juegos:

- Carecen de una historia central.
- Diseños poco elaborados en la mayoría de los juegos.
- Limitación con el idioma.

Con los aspectos antes analizados de los video-juegos homólogos, se puede constatar, que ninguno de los video-juego estudiados se ajusta al enfoque educativo tratado en el video-

juego “Recicla conmigo”, por este motivo se hace necesario el desarrollo del mismo, tomando en cuenta los aspectos positivos antes vistos.

1.3 Metodología de desarrollo

Las metodologías en el desarrollo de software, definen un conjunto de procedimientos y técnicas que ayudan a documentar y guiar el proceso. De esto se deriva la posibilidad de definir los roles, las fases y los implicados en cada una de las etapas de desarrollo. Estas teniendo en cuenta el nivel de documentación, de planificación y de control sobre los procesos, se clasifican en dos tipos principales (8; 9):

- **Robustas/Tradicionales**: Cuentan con un riguroso control de los procesos en el proyecto. Su principal característica es que se opta por planificar completamente el proyecto para a posteriori comenzar el desarrollo. Tienen como principal desventaja, que al ser planificado el proyecto integro al comienzo, es complicado ajustar el proyecto cuando surgen cambios inesperados durante el desarrollo y además implica un gran gasto en conceptos de planificación (8).
- **Ágiles/Ligeras**: Basan la estructuración de los procesos de manera dinámica, permitiendo de esta forma un patrón adaptable con la capacidad de ahorrar en tiempo cuando ocurren cambios en los requisitos del proyecto. El desarrollo bajo una metodología ágil tiene como pauta: la simplicidad, a mayor simplicidad en el desarrollo, menos peligro de demoras ante cambios. Debido a esto el desarrollo es incremental, dado que mayormente se utiliza la refactorización (aumentar la legibilidad del código, así como optimizarlo en caso de ser necesario) (8).

Para el desarrollo del presente trabajo se definieron las metodologías ágiles como las más acorde para ser utilizadas, puesto que dadas sus características, pueden brindar un mayor campo de acción en un proyecto pequeño, sujeto a cambios y carente de ciertos procesos que solo serían factibles en un proyecto de mayor envergadura.

1.3.1 Metodología “Scrum”

La metodología *Scrum* es una metodología de desarrollo de software que se basa en el enfoque ágil. Con *Scrum* se pueden realizar productos de forma rápida y organizada, puesto que distribuye el trabajo de manera tal que en el proceso de producción solo intervienen los desarrolladores, con baja participación del líder. Para garantizar la organización, *Scrum* define tres roles principales (10):

- **Product Owner**: Es en esencia el líder del proyecto. Dicho personaje no interviene directamente en el desarrollo del producto, ya que es el encargado de definir las funcionalidades del mismo al comienzo del proyecto, y cada cierto tiempo (que se define por parte del equipo de trabajo) se reúnen para analizar el avance de la aplicación en cuestión. Sobre él recaen también, las tareas de interactuar con los clientes y el chequeo de la buena comunicación por parte del colectivo de desarrolladores (11).
- **Scrum Master**: Es uno de los programadores del proyecto (generalmente el de más experiencia) y se encarga de realizar las reuniones diarias, en las que se analiza el avance por parte de cada uno de los programadores y se identifican los posibles problemas (dígase atrasos, falta de recursos o desconocimiento en ciertos temas de los desarrolladores) para a partir de ahí definir cómo solucionarlos y continuar en el proceso de desarrollo. El *Scrum Master* no se encarga de dirigir ni de orientar al equipo, este solo supervisa el proceso e informa al *Product Owner* de cualquier problema (11).
- **Scrum Team**: Colectivo de programadores que desarrollan el proyecto. El *Scrum Master* pertenece a este colectivo (11).

El funcionamiento de la metodología *Scrum,* se rige por los siguientes pasos:

- **Creación del Product Backlog (Producto acumulado):** Es el listado de funcionalidades del software. Este listado es creado por el *Product Owner* (líder del producto) después de reunirse con el cliente y haber discutido las cuestiones principales del proyecto. Estas funcionalidades se ordenan por prioridad y pueden verse sujetas a cambios a medida que avanza el proyecto (11).
- **Sprint Planning Meeting:** Es una reunión que se realiza el primer día de trabajo en el proyecto. En esta reunión se definen las funcionalidades de mayor prioridad y se pronostica cuánto tiempo puede tomar desarrollarlas. Una vez que se definieron estas funcionalidades y el tiempo para realizarlas, se confecciona un nuevo listado conocido como *Sprint Backlog* con esas funcionalidades (11).
- **Daily Scrum Meeting:** Son reuniones diarias que se realizan antes de empezar el día laborar (no más de 15 minuto), dirigidas por el *Scrum Master* (11).
- **Sprint Review:** Es una reunión que se realiza al finalizar el plazo de tiempo estimado para el *Sprint Backlog*. Aquí se reúnen el *Product Owner*, el *Scrum Master* y el *Scrum Team* para analizar el producto y ver si se han cumplido las metas (11).

- **Sprint Restropective:** Reunión que se realiza inmediatamente después del *Sprint Review*, con el objetivo de hacer un análisis del trabajo en ese período y evaluar la productividad de los desarrolladores. En este punto se vuelve a comenzar el ciclo nuevamente, con el *Sprint Planning Meeting* hasta completar el producto final (11).

Dentro de las principales desventajas con que cuenta Scrum, se pueden encontrar las siguientes:

- Dificultad para ser aplicado en proyectos de gran envergadura.
- Prácticamente no se genera ninguna documentación durante el proceso de desarrollo.
- El cliente debe participar de forma activa en el proyecto.
- Se requiere un experto de la metodología que supervise el proceso.

1.3.2 Metodología "Programación Extrema (XP)"

La metodología XP es una metodología ágil, que define una estrategia para desarrollar proyectos pequeños que se vean sujetos a cambios. Esta tiene un carácter incremental que por medio de una estrecha relación entre desarrolladores y cliente, permite crear el proyecto de forma gradual y chequear en todo momento si se están cumpliendo las expectativas del interesado. El flujo de trabajo con XP para el desarrollo de un proyecto, tiende a ser separado en cuatro partes importantes que definen la realización de software. (9)

1. **Planificación y análisis**: Se planifica y definen las prioridades de las tareas que van a ser desarrolladas en cada una de las iteraciones del proyecto (9).
2. **Diseño y codificación**: Como esta metodología se basa en un trabajo incremental, el diseño y la codificación van de la mano. En cada una de las iteraciones del proyecto van perfeccionándose estas cuestiones, siempre con la supervisión del cliente. Esta fase es la más extensa y se realiza en forma de iteraciones, donde en cada iteración se van realizando de forma incremental las especificaciones del proyecto (9).
3. **Pruebas del software**: Las pruebas de software se hacen en todo momento, y son principalmente de dos tipos (9):
    - Unitarias: Por parte de los desarrolladores para chequear el correcto funcionamiento de cada módulos.
    - Aceptación: Por parte del cliente, para ver si el proyecto cumple con las expectativas hasta ese momento del desarrollo.

4. **Despliegue**: Una vez que el proyecto culmina todas las fases anteriores y ha pasado las pruebas de software, está listo para ser entregado al cliente (9).

Las relaciones interpersonales es uno de los puntos más enfocados en esta metodología, con el objetivo de lograr un buen flujo de trabajo entre las distintas partes vinculadas al proyecto (líder, desarrolladores y cliente), y de esta forma incita y fomenta cinco valores principales que son: la simplicidad, la comunicación, la retroalimentación, el coraje y el respeto. Si se toman en cuenta cada uno de estos valores, las probabilidades de lograr exitosamente la finalización del software son grandes.

1.3.3 Metodología "Huddle"

Es una metodología ágil enfocada directamente al desarrollo de video-juegos, que se basa en Scrum y *Waterfall Process* (cascada). Huddle tiene como principal objetivo unificar un proceso en el cual se agrupen las principales acciones que se realizan en el desarrollo de un video-juego, dada las características especiales con que estos cuentan, tales como el dinamismo de los requisitos, que pueden llevar a una infinidad de posibles resultados que imposibiliten el hecho de representar uno o varios procesos (12).

Se separa en tres etapas principales durante el desarrollo de un video-juego:

Imagen 4: Proceso de desarrollo Huddle

Preproducción: Es donde se realiza el análisis y diseño del video-juego por medio de los siguientes documentos (12):

- Documento de diseño (guion técnico): se recopila toda la información necesaria para el desarrollo del video-juego, tales como: historia, personajes, niveles, entre otros. Este documento a lo largo del proceso de desarrollo puede verse sujeto a cambio, producto de la evolución misma del proyecto.

- *Sprint Plan*: se hace un listado de *Sprint* que definen el proceso de desarrollo. Los *Sprint* se pueden tratar como iteraciones en el proceso de desarrollo de Huddle y como tal, cada uno representa una evolución con relación al anterior.
- *Feature Log*: son funcionalidades específicas identificadas en cada *Sprint* del proyecto. La terminación de los *Feature Log* implican el nivel de terminación del *Sprint* en cuestión al que pertenecen.

Producción: Se desarrollan las funcionalidades y características definidas en el diseño del proyecto. La guía a seguir en este proceso está definida por el listado de *Sprint* y a su vez por los *Feature Log* implicados a cada uno de ellos. En este proceso se realizan varias iteraciones que están ligadas a la cantidad de *Sprint* con que cuenta el *Sprint Plan* (12).

Una vez culminada la fase de desarrollo definida en el *Sprint Plan*, se pasa a la fase de prueba. El objetivo es ver los posibles errores y si el video-juego cumple con los requerimientos del diseño, en caso de encontrarse algún problema se pasaría nuevamente al proceso de Preproducción y se comienza otro ciclo de desarrollo a partir de los nuevos *Backlog* creados.

Postmortem: Esta fase se destina al análisis del proyecto, con vistas a discutir los aspectos positivos y analizar los aspectos negativos. De esta forma se persigue contribuir con la experiencia a futuros proyectos del equipo de desarrollo (12).

1.3.4 Metodología de desarrollo seleccionada

Para la selección de la metodología se tuvieron en cuenta los siguientes aspectos:

- Proceso ágil de desarrollo.
- Aplicable para el desarrollo de video-juegos.
- Proceso iterativo e incremental.

A partir de estos aspectos se confeccionó la siguiente tabla comparativa para apoyar dicha selección:

	Scrum	**XP**	**Huddle**
Proceso ágil	SI	SI	SI

Aplicable a video-juegos	Aplicable en el desarrollo de todo tipo de software.	Aplicable en el desarrollo de todo tipo de software.	Destinada para el desarrollo de video-juegos.
Proceso iterativo e incremental	SI	SI	SI

Tabla 1: Comparación de las metodologías

A partir de la tabla anterior se seleccionó la metodología Huddle, puesto que junto a XP y Scrum, puede ser aplicada en el desarrollo del video-juego “Recicla conmigo”, pero cuenta con la característica de ser destinada directamente al desarrollo de video-juegos, que en el caso de las otras dos opciones, llevaría un proceso de adaptación en la confección del diseño y de algunos procesos como la definición de los requisitos funcionales (ya que el dinamismo de los video-juegos dificultan esta tarea), además de que dicha metodología se encuentra aplicada actualmente en el centro VERTEX de la facultad 5, que con vistas a validar su uso en la universidad, cuenta con el documento titulado “Permiso de no utilización de artefactos”, que permite al centro hacer uso de los artefactos definidos por Huddle durante el proceso de desarrollo de video-juegos.

1.4 Herramientas utilizadas

1.4.1 Lenguaje de desarrollo

Java es un lenguaje de programación concurrente y orientado a objetos, diseñado para ser ejecutado en cualquier dispositivo sin la necesidad de ser recompilado, esto gracias a que las aplicaciones en Java, son ejecutadas por una plataforma (la plataforma es la que debe estar instalada en el dispositivo) que es la encargada de interpretar la aplicación y ejecutarla. Siendo esta característica una de las mayores ventajas de este lenguaje, ha hecho que se encuentre dentro de los más usados en la actualidad para el desarrollo de software. Android como uno de los SO para móviles más usados, tiene como lenguaje nativo Java, convirtiéndolo en el lenguaje por excelencia de dicho SO (13).

1.4.2 Framework de desarrollo

Durante la investigación se consultaron diferentes *framework*, seleccionando de ellos el más compatible con las especificaciones del video-juego “Recicla conmigo”. En la selección del

framework se tuvo en cuenta, aspectos tales como: la comunidad existente en internet, soporte para juegos en 2 dimensiones (2D) y licencias de libre distribución. Las herramientas estudiadas fueron las siguientes:

LibGDX

Es un *framework* multiplataforma para el desarrollo de video-juegos en los sistemas operativos Windows, Linux, Android, MAC OS y iOS, aunque en este último SO es necesario utilizar monotouch (aplicación de desarrollo no gratuita). Está desarrollado bajo el lenguaje de programación Java, aunque algunas partes fueron desarrolladas en C/C++ para el trabajo en matemática y física (14).

Ventajas

- Vasta documentación, tutoriales y ejemplos.
- Actualizaciones contantes por parte de los desarrolladores.
- Manejo de audio, física, matemática y archivos.
- Soporta 2d y 3d.
- Posibilidad o permite exportar juegos realizados en Java con extensión .jar a Android con máxima compatibilidad.

Desventajas

- El soporte 3D aún no se encuentra completamente funcional.

AndEngine

Es un *framework* libre creado por Nicolas Gramlich para el desarrollo de video-juegos 2D en Android. Este *framework* está desarrollado en Java y cuenta con una amplia comunidad alrededor del mundo. Por medio del mismo es posible el trabajo matemático y físico para las aplicaciones que se realicen y se puede vincular con Box (librería gratuita con una vasta gama de soportes físicos y matemáticos) para trabajos más avanzados de física (15).

Ventajas

- Soporte 2d.
- Manejo con audio, física y matemática.
- Gran cantidad de ejemplos en internet.

- Activa comunidad en los foros de la herramienta.

Unity

Este *framework* se encuentra en la actualidad dentro de los más usados para el desarrollo de video-juegos, dada la amplia gama de plataformas para las que permite desarrollar tales como: Windows, MAC OS, Android, iOS, entre otras (16).

Unity cuenta además con una versión libre, aunque la misma no brinda el gran cúmulo de herramientas y recursos que se pueden encontrar con la versión completa. El lenguaje de programación nativo de este *framework* es C/C++ pero para el desarrollo de video-juegos el usuario no necesita ser un experto en programación, ya que en la confección de los juegos se utilizan lenguajes *scripts* como Javascript, C# o Boo (dialecto de Python) (16).

Para la realización de un video-juego con Unity se utilizan las herramientas y editores con que cuenta el mismo, ya que trae editores de escenarios y su propio IDE para la lógica del video-juego (16).

Ventajas:

- Brinda una gran gama de herramientas y facilidades para el desarrollo de video-juegos.
- No es necesario tener un amplio dominio de programación.
- Cuenta con una versión libre que permite probar el software.

Desventajas:

- La versión libre no cuenta con todas las funcionalidades y recursos disponibles en las versiones de pago.
- Inicia por defecto con el logotipo de Unity, cosa que no todos los desarrolladores desean en sus juegos.
- La versión gratis no cuenta con la optimización de módulos, por lo cual las aplicaciones creadas con esta versión, tienden a ocupar mayor espacio de almacenamiento.

Selección del framework

Luego de un análisis detallado de cada una de las herramientas antes estudiadas, se define la siguiente tabla comparativa para llegar a la conclusión de cuál de ellas utilizar como base

para el desarrollo del video-juego “Recicla conmigo”, a partir de los siguientes puntos de comparación.

- Plataforma(s) de desarrollo del framework.
- Gráficos permitidos por el framework.
- Lenguaje de programación para desarrollar con dicho framework.
- Tipo de herramienta (a partir de las prestaciones que brinda).
- Tecnología utilizada por el framework.
- Licencia de distribución del framework.

	AndEngine	**LibGDX**	**Unity**
Plataforma	Android	Android, iOS, Windows, Linux, entre otros.	Android, iOS, Windows, entre otros.
Gráficos	2D	3D	3D
Lenguaje de programación	Java	Java	C#, Javascript, Boo
Tipo	*Game Engine*	*Framework*	*Game Engine*
Tecnología	*OpenGL* ES 1.0 y *2.0*	*OpenGL* ES 1.0 y 2.0	OpenGL ES (Android y iOS)
Licencia	LGPL	Apache 2.0	GPL

Tabla 2: Comparación de los framework

A partir de la tabla anterior, se descartó el framework Unity por los siguientes motivos:

- Uso de gráficos 3D, no necesarios puesto que el video-juego “Recicla conmigo” fue desarrollado con gráficos 2D. Esto trae consigo la adaptación de una tecnología en otra, complejizando el desarrollo.
- Con vistas a facilitar el desarrollo se definió utilizar el lenguaje de programación Java, siendo el mismo el lenguaje de programación nativo de Android, característica que no cumple Unity, puesto que los lenguajes de desarrollo que utiliza son C#, Javascript y Boo.

- Como punto definitivo en la exclusión de dicho framework, está la licencia de uso. Cuenta con 2 licencias de uso, i) Unity Personal y ii) Unity Pro, de las cuales la versión Pro es de pago por tanto no es posible su uso, mientras que la versión Personal es libre y puede utilizarse en el desarrollo de video-juego con fines comerciales, pero con la restricción de que la institución o persona que lo comercialice, no puede tener un ingreso o capital mayor que USD 100,000 en el último año fiscal, lo que traería consigo la imposibilidad de comercial el software desarrollado, fuera del ámbito nacional (15).

Quedando la selección entre AndEngine y LibGDX. De estos framework se eligió AndEngine, por cumplir con todos los aspectos definidos para la selección del framework, descartando a LibGDX, principalmente por considerarse un Framework más que un Game Engine, puesto que LibGDX se enfoca más en las prestaciones multiplataforma que brinda, que en las funcionalidades para facilitar el desarrollo del video-juego (abstrayendo al programador del lenguaje de bajo nivel), además de contar con gráficos 3D (visto en Unity), que aunque no es un punto de peso, puede significar una posible demora en el desarrollo.

1.4.3 Entorno Integrado de Desarrollo (IDE)

En la actualidad Android Studio es el IDE por excelencia para el desarrollo en SO Android, dado que Google ha apostado por esta nueva herramienta dejando de lado a herramientas como Eclipse que fue de las primeras utilizadas. Android Studio cuenta con una gran cantidad de prestaciones y recursos que facilitan el desarrollo, ejemplos de esto son:

- Refactorización del código (nativa).
- Renderización en tiempo real.
- Plantillas para la creación rápida de aplicaciones.

La licencia bajo la que se encuentra Android Studio es Apache 2.0, que es una licencia de software libre y código abierto que permiten el uso del mismo sin problemas legales fuera del ámbito de Cuba (17).

1.4.4 Sistema gestor de base de datos

SQLite

Es un motor de base de datos SQL incorporado, que a diferencia de otros motores de base de datos conocidos, como PostgreSQL o MySQL (que necesitan de un servidor independiente

que brinde los servicios a las aplicaciones por medio de la red u otras vías), lee y escribe directamente en un archivo en el disco duro de la computadora. El archivo generado por la base de datos (.db) puede ser fácilmente exportado de una máquina a otra y de un sistema a otro sin presentar problemas de compatibilidad (18).

SQLite Browser

Es un ligero GUI (Interfaz gráfica de usuario) para la edición de base de datos SQLite. Este software está desarrollado en Qt y se encuentra bajo licencia GPL, con su código compartido en los repositorios del GITHUB, disponible para todos los interesados en contribuir a su desarrollo. Con SQLite Browser es posible crear, modificar y editar base de datos SQLite por medio de una interfaz amigable e intuitiva para el usuario (19).

Por medio de este GUI se van a realizar la base de datos del video-juego “Recicla conmigo”, donde se almacenarán todos los datos relacionados a los niveles haciendo más dinámico el juego y permitiendo que en futuras versiones sea posible la inserción de nuevos niveles sin tener que modificar el código fuente del video-juego.

1.4.5 Herramienta de modelado UML

Visual Paradigm es una herramienta CASE que se utilizada para el modelado UML. La misma es de libre distribución y se caracteriza por confiabilidad y estabilidad, por este motivo, es la herramienta perfecta para Ingenieros de Software, Analistas de Sistemas, entre otros, que necesiten desarrollar bajo un sistema UML orientado a objetos (20).

1.4.6 Diseño y edición de imágenes

Edición de imágenes

La aplicación GIMP es un editor de imágenes multiplataforma disponible para GNU/Linux, OS X, Windows, entre otros sistemas operativos. GIMP es un software libre, por tanto es posible tener acceso a su código fuente y ser cambiado a gusto del usuario, teniendo el derecho incluso de su redistribución bajo los nuevos cambios realizados. Este software provee de una infinidad de sofisticadas herramientas para la edición de imágenes pudiendo ser utilizado por fotógrafos, diseñadores, ilustradores y demás personas que lo necesiten. En el presente trabajo se utilizará para el retoque, ajuste y diseño de las imágenes del video-juego (21; 22).

Edición de gráficos vectoriales

Como editor profesional de gráficos vectoriales se utilizó Inkscape, que se encuentra disponible para la mayoría de los SO conocidos. Con esta herramienta es posibles realizar desde simples trabajos con imágenes, hasta complejos maquetados para páginas web o para la producción de interfaces de software. En el desarrollo del video-juego “Recicla conmigo”, se utilizará para el maquetado de las escenas, así como del posicionamiento de las imágenes, en los atlas de memoria que utilizan los *Game Engine* (como es el caso de *AndEngine*) (23).

Conclusiones parciales

Con el presente capítulo se arribó a las siguientes conclusiones parciales:

- Se estudiaron los video-juegos homólogos al juego “Recicla conmigo”, pudiéndose constatar que era necesario el desarrollo del mismo.
- La herramienta AndEngine fue la elegida para el desarrollo del video-juego, puesto que brinda los recursos necesarios que facilitan el proceso de desarrollo.
- Se definió la metodología Huddle como metodología para guiar el proceso de desarrollo, puesto que de las estudiadas fue la que más se ajustó a las especificaciones del presente trabajo.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL VIDEO-JUEGO

Con el análisis y diseño del video-juego se persigue definir una estructura que más tarde en el desarrollo, guíe el proceso. La metodología Huddle cuenta con una serie de artefactos que facilitan el análisis y diseño, tales como el documento de diseño y el listado *Sprint Plan*. Con el documento de diseño se esboza el video-juego, a partir de todos los datos relevantes del mismo, de manera tal que una vez finalizado el guion, en este quede el concepto de lo que se desea desarrollar, mientras que con el listado *Sprint Plan*, se van a definir las iteraciones necesarias a llevarse a cabo durante el desarrollo. Cada iteración, es una versión del producto y debe implicar un avance progresivo con relación a la iteración anterior.

2.1 Documento de diseño del video-juego

Concepto	
Título	Recicla conmigo
Estudio/Diseñadores	Guillermo González Jiménez
Plataforma	Sistema operativo (SO) Android
Versión (documento)	1.0
Sinopsis del modo de juego y contenido	Recicla es un video-juego destinado a niños de la enseñanza primaria. Con este juego se persigue inculcar en los jóvenes y niños la importancia del reciclado, así como identificar los tipos de basura y clasificarlos para su reciclado.
Categoría	Educativo
Licencia	Licencia libre
Mecánicas	El juego se basa en la recolección de desechos en la escena del nivel que se esté jugando en ese momento. El usuario debe encontrar los desechos en la escena y arrastrarlos hacia el contenedor correspondiente. Existen 4 tipos principales de desechos: papel, plástico, metal y vidrio,

	además del correspondiente contenedor para cada uno de ellos. En los primeros niveles solo se recoge un tipo de desecho y a medida que avanza el juego se van desbloqueando nuevos tipos de desechos.
Tecnología	**Hardware**: • Medio: Smartphone • Memoria: Mínimo (64 Mg de RAM) Recomendado (128Mg RAM) • Almacenamiento: Mínimo (30 Mg) Recomendado (100 Mg) **Software** • Sistema Operativo: Android 2.0 o mayor • Lenguaje de programación: Java • Marco de trabajo: AndEngine
Público	Principalmente a niños entre 6 y 12 años de edad.
Objetivo pedagógico	Se persigue que los niños desarrollen habilidades a medida se completan los diferentes niveles del video-juego.
Habilidades	Lectura y compresión: inicialmente se muestra la historieta del juego donde el niño debe leer e interpretar, además de otros textos en forma de mensajes que aparecen durante el juego como son las explicaciones del nivel de entrenamiento y las descripciones de cada nivel. Desarrollo de habilidades motoras: para jugar es necesario encontrar y arrastrar cada desecho que se encuentra en el escenario.

Historia de versiones			
Versión	**Autor**	**Descripción**	**Fecha**
1.0	Guillermo González Jiménez	Versión inicial	29/03/2016
2.0	Guillermo González Jiménez	Rectificación de errores en el documento de diseño.	14/06/2016

Visión general del juego	
Progresos del juego	• Por cada nivel se mostrará una imagen de la escena del nivel. • Sobre la imagen de la escena se muestran los desechos reciclables aleatoriamente distribuidos. • En lugares fijos se muestran objetos del entorno, que pueden ser animados o estáticos. • En lugares fijos se muestran los contenedores, con sus logos que identifican el tipo de desecho que almacena.
Estructura de las misiones	Por nivel el usuario debe ser capaz de encontrar cada uno de los desechos y colocarlos en sus respectivos contenedores para su reciclaje.
Objetivos	El objetivo del juego consiste en reciclar correctamente todos los desechos que se encuentran en cada una de las escenas. Mejorar las habilidades motoras de los niños.
Flujo del juego	Primeramente se muestra al usuario la historia del juego, con una serie de imágenes continuas que definen el hilo principal del juego. Seguidamente se va transitando por

<table>
<tr><td></td><td>cada uno de los niveles, los cuales van complementando dicha historia.

Cada uno de los niveles una vez que son comenzados, muestra un mensaje donde se encuentra la descripción del nivel, siempre siguiendo el hilo del juego. Una vez que se culmina un nivel se muestra un mensaje resumen con las estadísticas del nivel completado y se carga el siguiente nivel, en caso de ser el nivel final entonces se muestra un mensaje de felicitación con las estadísticas generales alcanzadas.</td></tr>
<tr><td colspan="2">Mecánicas del juego</td></tr>
<tr><td>Cámara</td><td>La cámara es usada bajo una resolución de 960 píxeles (px) de ancho por 540 px de alto. Esta resolución tiene un ratio que oscila en la media de las resoluciones de la mayoría de los dispositivos móviles actuales. Todos los gráficos del video-juego son en 2 dimensiones (2D).</td></tr>
<tr><td>Controles</td><td>Para la interacción con el juego solo es necesario hacer uso del panel táctil (touchpad) del dispositivo.</td></tr>
<tr><td>Puntajes</td><td>1. Al comenzar cada nivel se brinda un punto de lupa (las pistas) al usuario. Si al finalizar el nivel no es usada la lupa, el usuario gana 5 puntos extras.
2. Por cada elemento reciclado se ganan puntos acorde al tipo de elemento, a partir la siguiente tabla:
<table>
<tr><th>Tipo de desecho</th><th>Puntos</th></tr>
<tr><td>Papel</td><td>1 punto</td></tr>
<tr><td>Plástico</td><td>2 puntos</td></tr>
<tr><td>Vidrio</td><td>3 puntos</td></tr>
<tr><td>Metal</td><td>4 puntos</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td>3. Según el tiempo necesitado para completar el nivel, el usuario puede ganar puntos extras en el juego, estos están basados en la siguiente tabla de puntuaciones:
<table><tr><th>Tiempo empleado</th><th>Puntos</th></tr><tr><td>Menos de 1 min</td><td>20 punto</td></tr><tr><td>Menos de 2 min</td><td>15 puntos</td></tr><tr><td>Menos de 3 min</td><td>10 puntos</td></tr><tr><td>Menos de 4 min</td><td>7 puntos</td></tr><tr><td>Menos de 5 min</td><td>5 puntos</td></tr></table></td></tr>
<tr><td>Guardar/Cargar</td><td>El juego no cuenta con un sistema manual de guardado o cargado, este proceso se ejecuta de forma automática cuando el usuario cierra la aplicación, de esta forma solo se guardan los juegos que son cerrados por el usuario. Al comenzar nuevamente el juego por el usuario, se le pregunta si desea continuar la partida guardada o comenzar una nueva.</td></tr>
<tr><td>Movimientos</td><td>Desechos: Estos objetos son controlados por el usuario, ya que deben ser arrastrados hasta sus correspondientes contenedores.</td></tr>
<tr><td>Acciones</td><td>Activar lupa: Al activar la lupa, se indica al usuario donde está uno de los recursos (de forma aleatoria) que le faltan por encontrar en el nivel.

Activar/Desactivar guante: Al activar el guante se indica al usuario que es seguro recoger la basura y se puede pasar a buscarla por el nivel, en caso de estar desactivado no se permite arrastrar los desechos.</td></tr>
</table>

	Desechos: Los recursos se pueden arrastrar hacia los contenedores si se cumplen alguna de las siguientes acciones: • No hay mensajes en pantalla. • Esta activado el guante para recoger basura. **Contenedores**: Los contenedores activan sus respectivas animaciones cuando un desecho es arrastrado encima del mismo.

Estados del juego

Un estado del juego no es más que un momento en el video-juego accesible por el usuario. Ejemplos de estados pueden ser: encontrarse en el menú del juego, estar en la historia del juego o encontrarse jugando uno de los niveles. Los estados se enlazan por eventos que pueden ser condicionales o dirigidos, como es el caso de la transición menú-historia o menú-juego, donde se generan dos posibles caminos en la transición de los estados.

Para el modelado de un diagrama de estados se utilizan 5 elementos básicos:

- **Estado inicial**: Representa el inicio del diagrama y se simboliza con un círculo relleno.
- **Estados intermedios**: Son los estados del diagrama, y en estos se definen el nombre del estado, las variables de estado y las acciones del mismo, generalmente se simbolizan con un rectángulo redondeado.
- **Estado final**: Representa el fin de los procesos en el diagrama y pueden existir varios en el diagrama, se representan con un círculo sin relleno.
- **Transiciones**: Es el cambio de un estado a otro, pueden ser automáticas, condicionadas o manuales y se representan con una flecha.
- **Condición**: Se refieren a condicionales que se pueden encontrar en la secuencia de transiciones entre los estados que se ven sujetas a diferentes valores o elecciones tomadas por los usuarios o automatizadas por el software.

En el proceso de diseño del video-juego “Recicla conmigo”, se confeccionó el siguiente diagrama de estado, para representar cada uno de los estados posibles del video-juego,

así como las transiciones existentes en el mismo que permite el dinamismo y la fluidez del juego.

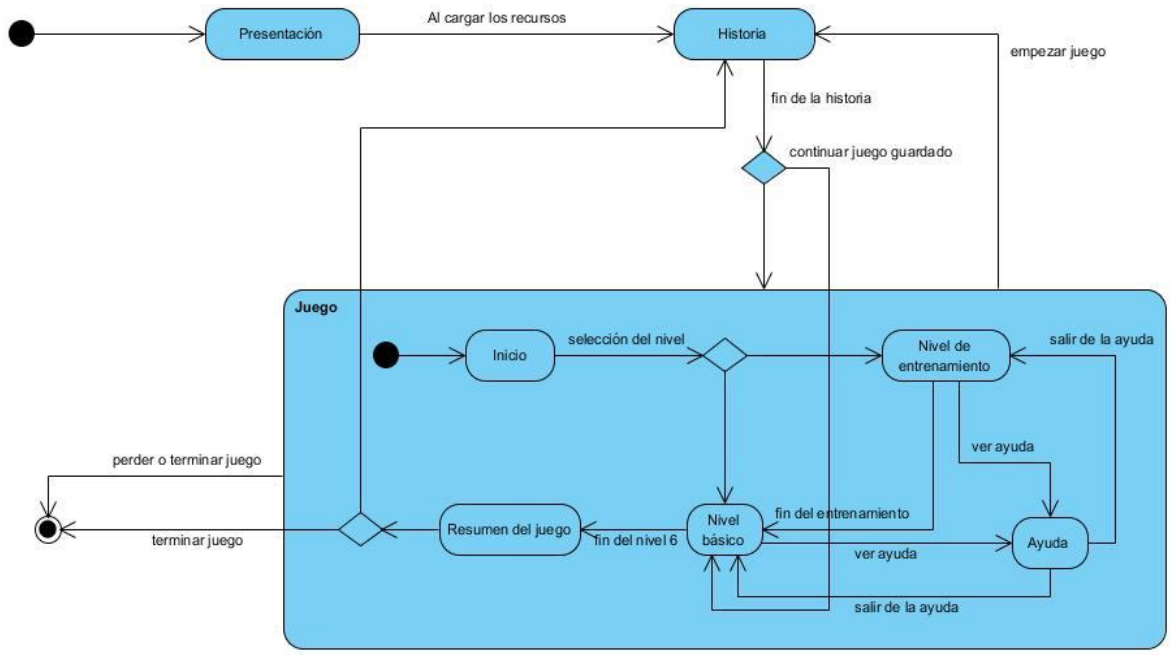


Imagen 5: Diagrama de estado del video-juego

Interfaces

Presentación del juego (splash)

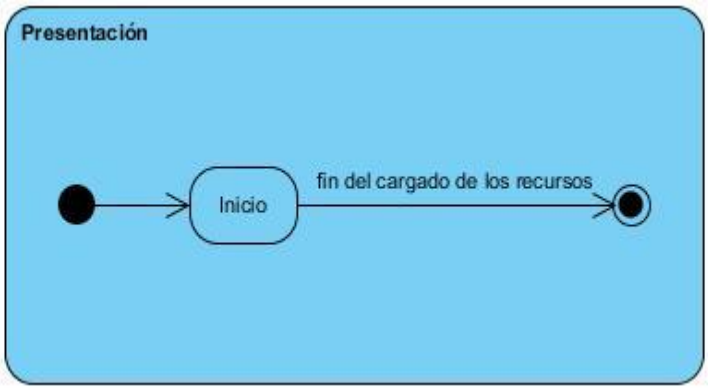
Descripción de pantalla	Se muestra el logotipo de la colección **Mundoclick** y contiene una barra de progresos que lleva el porciento de cargado en memoria de las texturas del juego.
Estados del juego	

Imagen 6: Diagrama de estado de la interfaz presentación

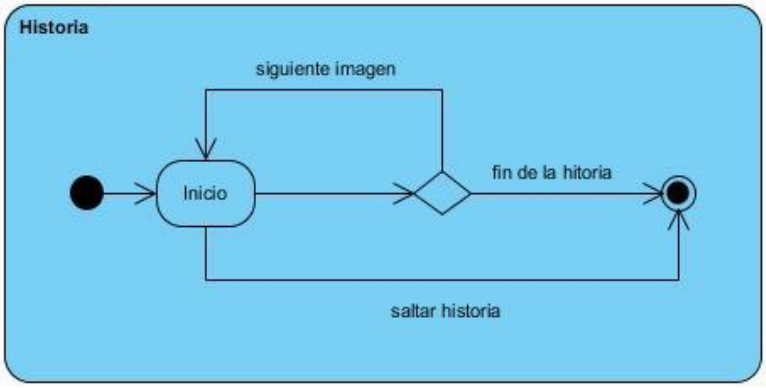
Imagen	 **Imagen 7: Escena de presentación**
Historia del juego	
Descripción de pantalla	En la historia inicial se debe mostrar un niño campesino que va de visita a la ciudad y ve la contaminación que tienen las calles; este niño decide formar un grupo con sus amigos de la ciudad para recoger basura y dejar la ciudad limpia.
Estados del juego	 **Imagen 8: Diagrama de estado de la interfaz historia**

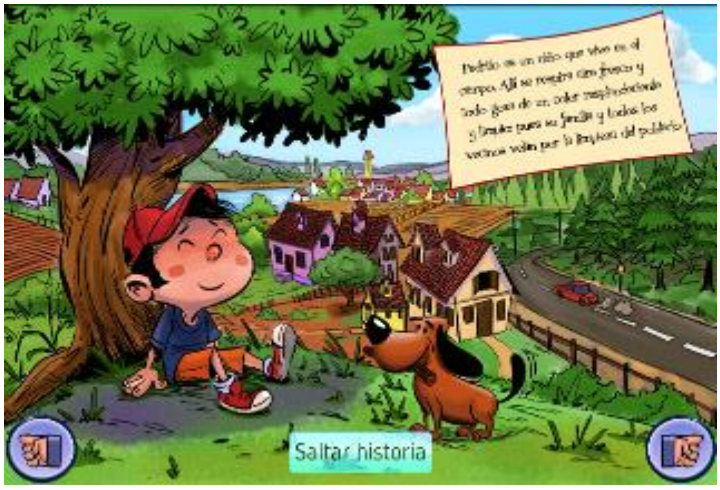
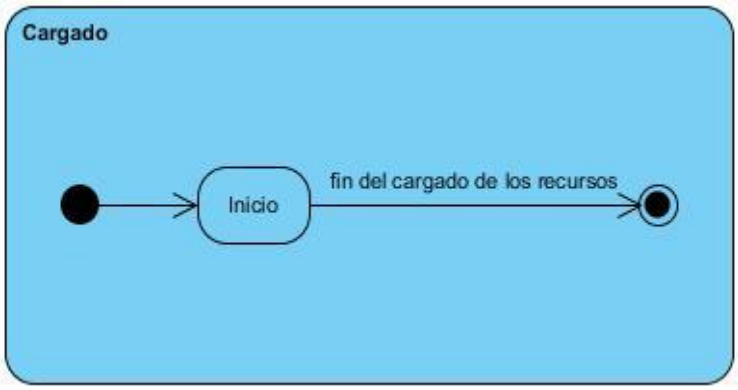
Imagen	 **Imagen 9: Escena historia**
Cargado (*loading*)	
Descripción de pantalla	Muestra una imagen animada, que significa que el juego se encuentra en ese momento cargando en memoria algún recurso. Esta escena es utilizada en el cambio de niveles, para de forma concurrente cargar el nivel y mostrarlo una vez que se culmine el proceso.
Estados del juego	 **Imagen 10: Diagrama de estado de la interfaz cargado**

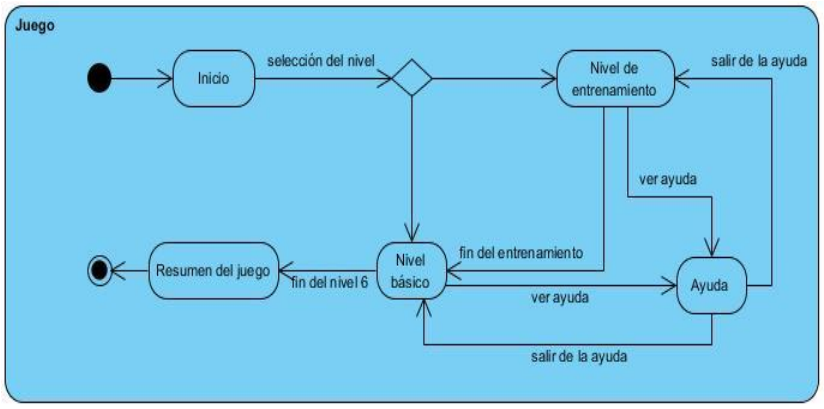
Imagen	 **Imagen 11: Escena de cargado**
Juego (*game*)	
Descripción de pantalla	Pantalla principal del juego en que se muestran los niveles. Inicialmente se pregunta si se desea jugar el nivel de entrenamiento o no. En caso de elegir el nivel de entrenamiento, se muestra dicho nivel y una vez completado se comienza en el nivel 1 del juego. En caso de no ser elegida la opción de nivel de entrenamiento, se comienza directamente en el nivel 1. Una vez que el último nivel es completado por el usuario, el juego comienza nuevamente en la historia para que el usuario lo intente una vez más.
Estados del juego	 **Imagen 12: Diagrama de estado de la interfaz juego**

Imagen	 **Imagen 13: Escena del juego**
Niveles	
Nivel de entrenamiento	
Encuentro	Jugar este nivel es opcional.
Descripción	Enseñar al usuario a interactuar con el video-juego.
Objetivos	Explica al usuario como jugar el video-juego.
Progreso	Al terminarlo lleva al usuario al primer nivel.
Objetos	Libro, papel y periódico.
Música	Música definida para los niveles.
Nivel 1	
Encuentro	Al comenzar el juego o al terminar el nivel de entrenamiento.
Descripción	Nivel básico con desechos de papel.
Objetivos	Recoger los desechos de la escena.
Progreso	Al finalizar direcciona directamente al nivel 2.
Objetos	4 recursos de papel dispersos por la escena.

Música	Música definida para los niveles.
Nivel 2	
Encuentro	Al terminar el nivel 1.
Descripción	Nivel básico con desechos de papel y de plástico.
Objetivos	Recoger los desechos de la escena.
Progreso	Al finalizar direcciona directamente al nivel 3.
Objetos	Recursos de papel y de plástico.
Música	Música definida para los niveles.
Nivel 3	
Encuentro	Al terminar el nivel 2.
Descripción	Nivel con desechos de papel y plástico.
Objetivos	Recoger los desechos de la escena.
Progreso	Al finalizar direcciona directamente al nivel 4.
Objetos	Recursos de papel, plástico y vidrio
Música	Música definida para los niveles.
Nivel 4	
Encuentro	Al terminar el nivel 3.
Descripción	Nivel con desechos de papel, plástico y vidrio.
Objetivos	Recoger los desechos de la escena.
Progreso	Al finalizar direcciona directamente al nivel 5.
Objetos	Recursos de papel, plástico y vidrio

Música	Música definida para los niveles.
Nivel 5	
Encuentro	Al terminar el nivel 4.
Descripción	Nivel con desechos de papel, plástico, vidrio y metal.
Objetivos	Recoger los desechos de la escena
Progreso	Al finalizar direcciona directamente al nivel 6.
Objetos	Recursos de papel, plástico, vidrio y metal.
Música	Música definida para los niveles.
Nivel 6	
Encuentro	Al terminar el nivel 5.
Descripción	Nivel con desechos de papel, plástico, vidrio y metal.
Objetivos	Recoger los desechos de la escena.
Progreso	Al finalizar muestra el resumen de las puntuaciones del juego y permite volver a empezar el juego o terminar.
Objetos	Recursos de papel, plástico, vidrio y metal.
Música	Música definida para los niveles.
Progresos del juego	

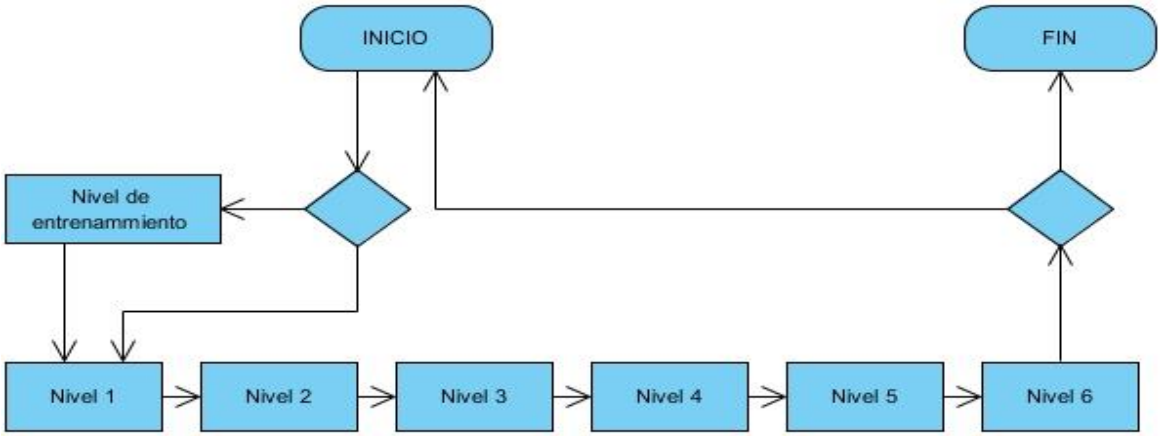


Imagen 14: Diagrama de secuencia del video-juego

Objetos	
Desechos	Son los objetos principales del juego, ya que es objetivo del juego encontrarlos y arrastrarlos hasta sus respectivos contenedores. Cada desecho se encuentra almacenado en la base de datos del juego, donde se guardan las dimensiones del mismo en el nivel, las posiciones (x, y) que ocupa, el tipo de desecho y la imagen del mismo.
Lupa	Es una ayuda que brinda el juego, se encuentra en la esquina inferior izquierda del escenario y de forma aleatoria, muestra al usuario donde se encuentra uno de los desechos en la escena.
Guante	Se encuentra al lado de la lupa y es preciso activarlo para recoger de forma segura los desechos.
Objetos de la escena	Los objetos son parte de la imagen de fondo, y permiten dar valor de profundidad al nivel, en muchos casos los desechos son escondidos detrás de estos objetos y de esta forma se dificulta la búsqueda de los mismos. Los objetos pueden ser de dos tipos: animados y estáticos. Cada uno de los objetos se encuentra almacenado en la base de datos donde se guardan las dimensiones en la escena, la posición (x, y) y la imagen del mismo. En el caso de los objetos animados se almacenan otros valores relacionados a la animación, como son: el tiempo de

	animación, la cantidad de columnas y filas que componen la animación.
Contenedores	Al igual que todos los objetos de cada nivel, este se almacena en la base de datos, pero es de un tipo diferente dado que al ser mostrado en las escenas son interactivos y semi-animados, ya que hasta ellos hay que arrastrar los objetos reciclables y una vez encima soltarlos. La animación de este objeto se activa en el momento en que se posiciona un objeto reciclable encima de él (animación de apertura del contenedor) o cuando se suelta el objeto (animación de cerrado).
Objetos en general	Todos los objetos cuentan con un atributo en común en la base de datos llamado zindex, este atributo define el nivel de profundidad de los objetos en la escena, de esta forma se determina que objetos van encima de otros al ser renderizado el nivel.
Opciones del juego	
Ayuda	
Cambios en el juego	Muestra el mensaje de ayuda del juego
Cambios en la mecánica	Congela el juego actual; se detiene el tiempo del nivel y se hace opaco todo excepto el mensaje.
Activado/Desactivado sonido	
Cambios en el juego	Cambia el ícono del audio y se activa el sonido, en dependencia del estado.
Cambios en la mecánica	Activado: se reproducen los sonidos del juego. Desactivado: no se reproducen los sonidos del juego.
Cerrar juego	
Cambios en el juego	Muestra un mensaje de confirmación para cerrar el juego.

Cambios en la mecánica	Congela el juego actual, se detiene el tiempo del nivel y se opaca todo excepto el mensaje.
Logros	
Si el usuario no hace uso de la lupa recibe al finalizar el nivel cinco puntos de bonificación.	

Tabla 3: Diseño del video-juego

2.2 Sprint Plan

Dentro de los principales artefactos generados por el proceso de desarrollo Huddle, se encuentran los *Sprint*. Cada *Sprint* define metas a cumplir y cada meta se desglosa en una serie de *Feature Log* que tributa al objetivo del *Sprint*. Para el desarrollo del proyecto, las tareas son ordenadas por prioridades y distribuidas entre los desarrolladores según sus conocimientos y capacidades, para agilizar el proceso. En cada momento del proyecto es posible ver el avance del mismo, dado que se va haciendo un seguimiento del estado en que se encuentran cada uno de los *Features Log* del *Sprint* en desarrollo, permitiendo tener control sobre todo el proceso (11).

Sprint 1: Flujo del video-juego

Flujo del Juego (*Game-flow*)	
Sprint ID	S01
Inicio	08/03/2016
Días	14
Fin	08/03/2016
Meta	Desarrollar las escenas del video-juego y relacionarlas por medio de los mecanismos definidos.

Tabla 4: Flujo del video-juego

Para la relación entre las escenas se confeccionó el siguiente diagrama de flujo (Imagen 15) que representa el avance del video-juego por las diferentes interfaces.

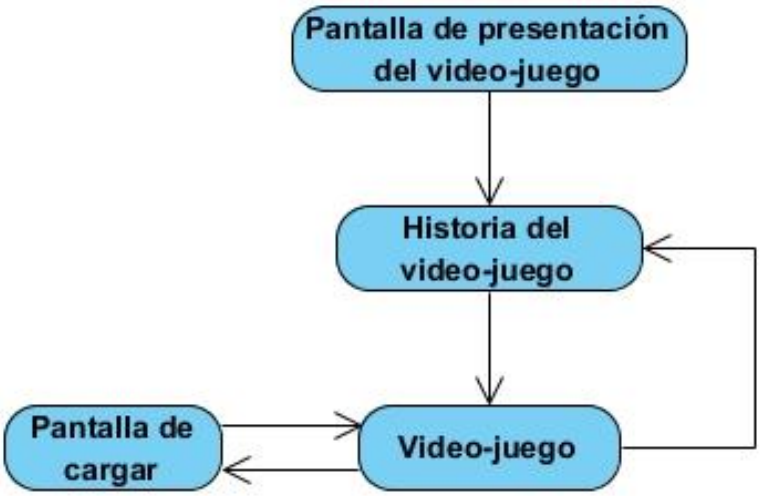


Imagen 15: Diagrama de secuencia de las escenas

Luego de ser creado el flujo de trabajo del video-juego se pasó al desarrollo de cada módulo independiente, así como la creación de las funcionalidades principales de la aplicación. Los *Feature Log* definidos en esta iteración, pueden ser encontrados en el Anexo 1.

Sprint 2: Funcionalidades principales del video-juego

Funcionalidades principales del video-juego	
Sprint ID	S02
Inicio	13/03/2016
Días	14
Fin	14/03/2016
Meta	Desarrollar los procesos independientes que se ejecutan en cada una de las escenas, que definen el comportamiento del juego.

Tabla 5: Funcionalidades principales del video-juego

Durante esta iteración se fue creando la infraestructura necesaria para el funcionamiento del video-juego, para facilitar el proceso de desarrollo en futuras iteraciones. Paralelo a esto se creó la infraestructura de clases necesarias para modelar el video-juego de forma eficiente, englobando las funcionalidades y sus relaciones. Como parte de este proceso se generó el siguiente diagrama de clases (Imagen 16) que define la estructura del software:

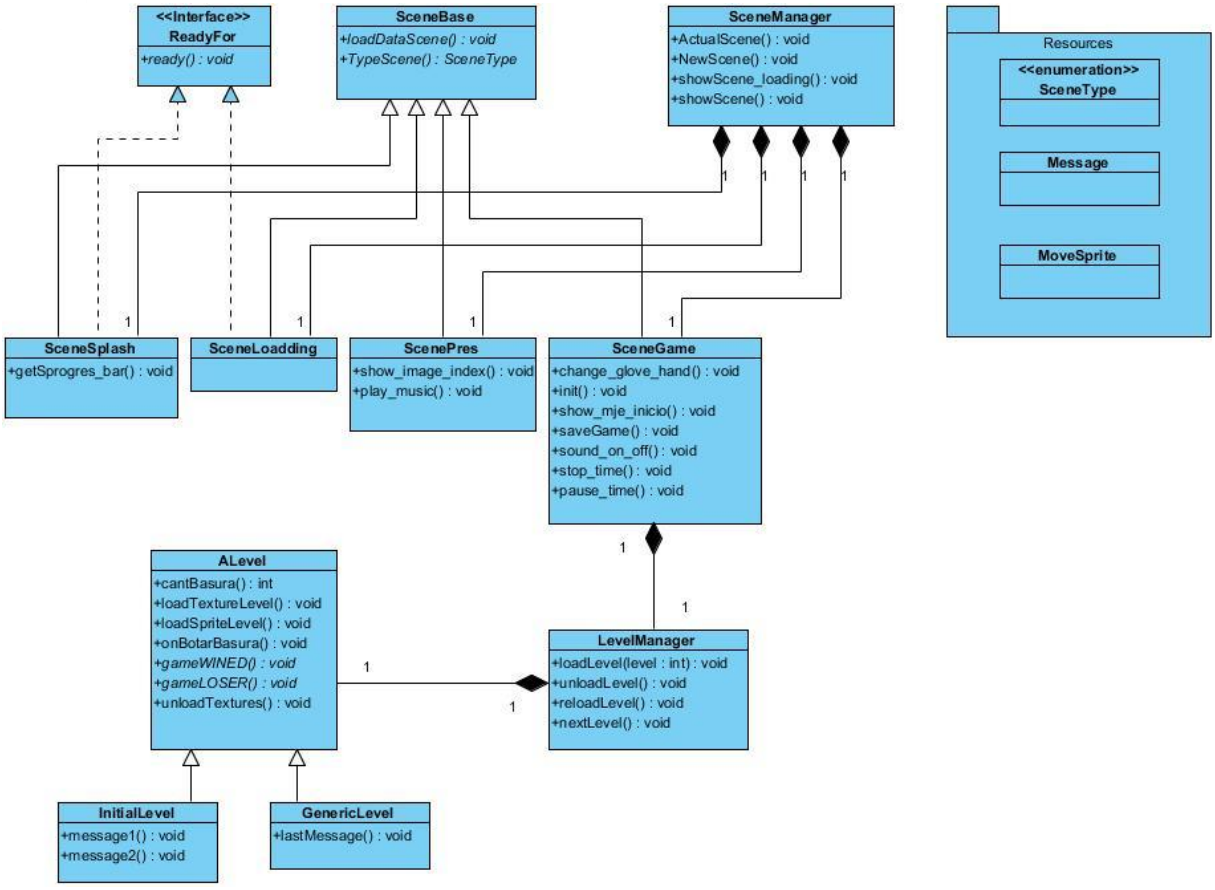


Imagen 16: Diagrama de clases del video-juego

Los *Feature Log* definidos en esta iteración, pueden ser encontrados en el Anexo 1 del documento.

Sprint 3: Base de datos del video-juego

Base de datos del video-juego	
Sprint ID	S03
Inicio	15/03/2016
Días	14
Fin	18/03/2016
Meta	Definir los niveles del video-juego dentro de la base de datos junto a los datos de configuración principales para mejorar el manejo de datos y la organización de la aplicación.

Tabla 6: Base de datos del video-juego

Durante la iteración actual se creó el modelo de datos (Imagen 17) del video-juego y las clases necesarias para la interacción con la base de datos del juego.

Imagen 17: Modelo de datos del video-juego

Luego de esto se pasó a crear la estructura de manejo de datos en la aplicación, que facilitara la relación con la base de datos (BD) y separará los procesos del software, de las consultas y operaciones en la BD. De esta forma se realizó el modelado de las clases pertenecientes a cada tabla de la BD, con lo cual se persigue que de forma automática se consulten los datos y estos se conviertan en un objeto manipulable por la aplicación. Se muestra a continuación el diagrama de clases para la lectura de la BD.

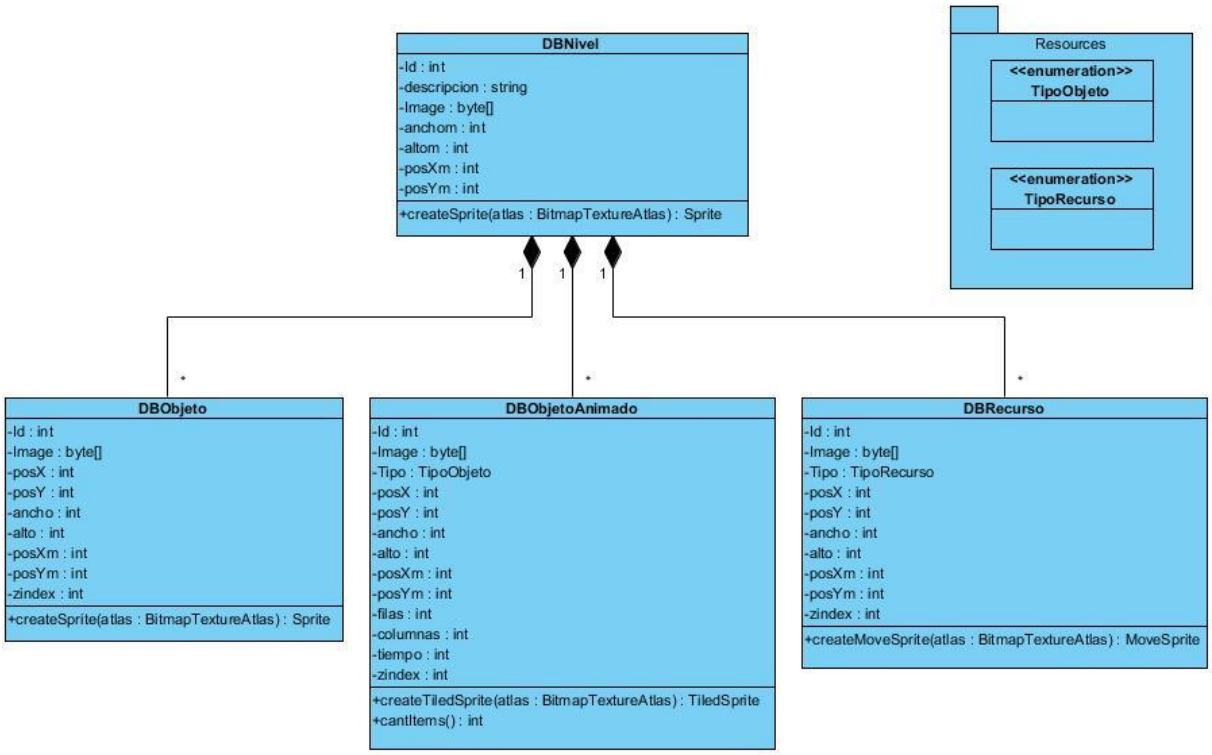


Imagen 18: Diagrama de clases para la BD

Cada clase modelada cuenta con un método ***create***, que varía en dependencia del objeto que se generará para el componente visual del video-juego. De esta forma se crea un modelo que actúa como facilitador de los datos, para el momento de ser utilizados. Para esta iteración los *Feature Log* definidos se pueden encontrar en el Anexo 1.

Sprint 4: Funcionalidades secundarias

Funcionalidades secundarias del video-juego	
Sprint ID	04
Inicio	20/03/2016
Días	14
Fin	08/05/2016
Meta	Desarrollo de las funcionalidades secundarias que permiten el manejo del juego y dan fin al proceso de desarrollo.

Tabla 7: Funcionalidades secundarias del video-juego

En el presente Sprint se persigue dar terminado al video-juego, agregando las funcionalidades menos importantes en el flujo del juego, pero que a su vez determinan el comportamiento del mismo. En el Anexo 1 se pueden encontrar los *Feature Log* definidos en el actual *Sprint*.

Dentro de los aspectos trabajados, se pueden encontrar algunos como el tratamiento de las puntuaciones del juego, las opciones del menú superior o el guardado/cargado de la aplicación, entre otros. De esta forma, una vez terminado este *Sprint* se termina parcialmente el desarrollo para pasar al proceso de prueba y corrección de posibles errores.

2.3 Arquitectura de software

La arquitectura seleccionada para el desarrollo del video-juego, fue N-Capas, dado que es la que más se ajusta a los requerimientos del software. La variante utilizada fue “3-capas” donde se desglosa el software en 3 niveles principales (24).

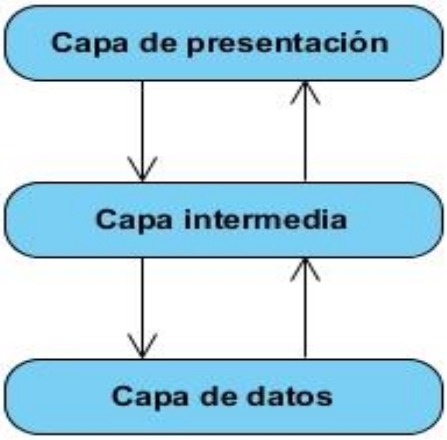


Imagen 19: Arquitectura N-capas

1. **Presentación**: Capa de interacción entre el usuario y la aplicación.

   Esta capa en el presente trabajo, se representa como las escenas del video-juego, que son las que permiten la interacción del usuario con el software.

2. **Intermedia**: Capa de enlace entre la capa de presentación y la capa de datos (es la capa lógica de la aplicación).

   La capa intermedia se definirá como las clases generadas para la conexión y trabajo con la BD, que permitan la comunicación de los datos con las interfaces.

3. **Datos**: Capa donde se tienen los datos físicos de la aplicación (base de datos).

   Los datos se representan como la BD que utilizará el video-juego, para almacenar la información necesaria de los niveles y demás valores funcionales del software.

2.4 Patrones de diseño

Un patrón de diseño es un cúmulo de experiencia agrupada por desarrolladores expertos, que con el tiempo definen una forma óptima para solucionar un problema determinado. El uso de patrones en el desarrollo de un software es altamente recomendado, puesto que aumentan la eficiencia de la aplicación y agilizan el proceso de desarrollo. Por tanto es posible decir que un patrón no es más que una dupla problema/solución y no surge a partir de una idea nueva, todo lo contrario; son el resultado de experiencia acumulada. En el campo de la informática los patrones pueden estar definidos como normas estructurales en el diseño del software o definir pautas a seguir respetando características valorativas (25). A continuación se muestran los patrones más importantes aplicados en el desarrollo del video-juego “Recicla conmigo”.

2.4.1 Singleton (clase única)

Este patrón garantiza que un objeto tenga una instancia única y a la vez tenga un punto de acceso global desde cualquier lugar de la aplicación (24). Un ejemplo de la utilización de este patrón en la aplicación es el siguiente:

```
public class ScenesManager {
    ...
    private static ScenesManager INTANCE=null;
    public static ScenesManager getIntance(){
        if(INTANCE==null){
            INTANCE=new ScenesManager();
        }
        return INTANCE;
    }
    ...
}
```

Imagen 20: Ejemplo del patrón Singleton

La clase **SceneManager** es la controladora del video-juego, y maneja todas las escenas y recursos, por este motivo es necesario tener una única instancia de la misma y que se encuentre disponible en cualquier momento.

2.4.2 Experto

Con la creación de un sistema informático se pueden llegar a definir de una a cientos de clases, y a las mismas se les pueden asignar de cientos a miles de responsabilidades. Durante

el proceso de diseño se determina la asignación de las responsabilidades a las clases definidas, y una buena asignación puede implicar la creación de un sistema fácil de entender, mantener y ampliar. Con el patrón **Experto** se persigue la correcta asignación de responsabilidades, definiendo las clases expertas que cuentan con la información solicitada por una funcionalidad determinada (25). Un ejemplo en la aplicación sobre este patrón se muestra en la siguiente imagen:

```
public class SceneGame extends SceneBase{
    //Para cambio de hand
    public void change_glove_hand(){...}
    //Inicio de la escena
    public void init(){...}
    //Para mostrar mensaje de inicio
    public void show_mje_inicio(){...}
    //Para salvar el game en la BD
    public void saveGame(){...}
    //Para activar/desactivar el sonido de la app
    public void sound_on_off(){...}
    //Parar el tiempo del nivel
    public void stop_time(){...}
    //Pausar el tiempo del nivel
    public void pause_time(){...}
}
```

Imagen 21: Ejemplo del patrón Experto

Como es posible apreciar, la clase **SceneGame** (Interfaz gráfica donde se muestran los niveles del juego) es la encargada de gestionar sus propias responsabilidades, como pueden ser: *stop_time* (parar el tiempo), *sound_on_off* (detener o ejecutar el audio), entre otras acciones nativas. Es así, como por medio del patrón **Experto**, se garantizó una correcta asignación de responsabilidades, aumentando el encapsulamiento (los objetos utilizan su propia información aumentando su independencia) que trae consigo la creación de clases sencillas y fáciles de entender (25).

2.4.3 Creador

Una de las operaciones más realizadas en una aplicación es la creación de objetos, por este motivo es necesario definir una buena práctica relacionada a ello, que permita una correcta asignación de dicha responsabilidad. El patrón **Creador** es aplicable cuando se cuenta con una clase A donde se tiene alguno de los siguientes casos (25):

- A agrega los objetos B.

- A contiene los objetos B.
- A contiene las instancias de los objetos B.
- A utiliza específicamente los objetos B.
- A contiene los datos de inicialización del dato B.

En cuyos casos se puede determinar que A es un creador de los objetos B.

La aplicación del patrón **Creador** en el presente trabajo, puede verse en práctica en la siguiente imagen:

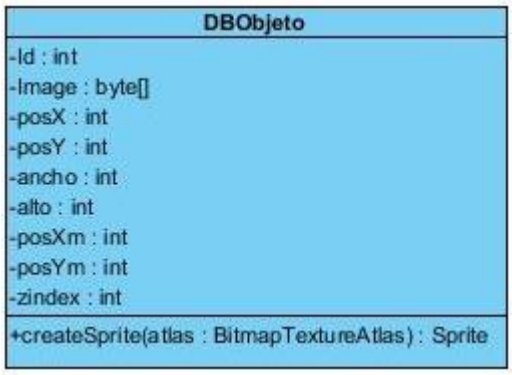


Imagen 22: Ejemplo del patrón Creador

Como es posible apreciarse en el diagrama de clases de la Imagen 22, se muestra la clase de conexión a la base de datos: **DBObjeto**, que cuenta con los atributos de los objetos visibles en los niveles, por este motivo es la encargada de crear el objeto Sprite (objeto a mostrarse en la escena) que se agregará a la escena al cargarse el nivel. De esta misma forma se encuentran estructuradas las clases **DBObjetoAnimado** y **DBRecurso**, aumentándose así la independencia de estas clases, que implica mayor probabilidad de reutilización y soporte de las mismas.

2.4.4 Controlador

Los eventos del sistema se consideran como eventos de alto nivel, y la captura de los mismos por parte de las clases creadas en la aplicación implicaría la inclusión de dependencias en las mismas, disminuyendo el encapsulamiento y simplicidad del código. Con el objetivo de resolver dichos problemas, se utiliza el patrón **Controlador**, que asigna la responsabilidad de los eventos del sistema a una clase controladora encargada de interpretarlos (25).

El patrón **Controlador** se aplicó en el presente trabajo en la clase **SceneManager**, como se puede ver en el diagrama de clases (Imagen 23), la misma se encarga del control de las

acciones relacionadas con las escenas, tales como la creación de nuevas escenas (**NewScene**) o de mostrar las escenas secundarias (**showScene_loading**) del video-juego. De esta forma se organizaron las responsabilidades del sistema sin llegar a saturar la clase controladora, problema común que puede traer consigo la dependencia entre clases y la falta de simplicidad del código.

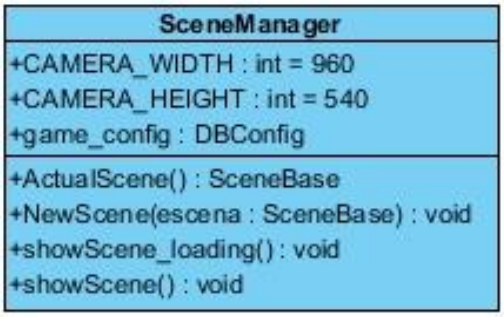


Imagen 23: Ejemplo del patrón Controlador

Conclusiones parciales

Con el desarrollo del presente capítulo se arribó a las siguientes conclusiones:

- El documento de diseño permitió esbozar el video-juego y definir el producto objetivo, haciendo posible aumentar la eficiencia y minimizar el trabajo durante el desarrollo.
- La arquitectura de software junto a los patrones de diseño utilizados permitirán desarrollar el video-juego siguiendo buenas prácticas que permitan facilitar la creación de nuevas versiones y solución de posibles errores.
- Los *Sprint* y los *Feature Log*, permiten guiar el proceso de desarrollo de forma eficiente y estructurada, minimizando el tiempo y definiendo las tareas a realizarse.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

Implementación y prueba es el último proceso que se lleva a cabo en el desarrollo de una aplicación. Con esta fase se hace de forma práctica, todo aquello que fue antes definido en las fases de investigación y análisis-diseño. La guía de este proceso es a partir de la metodología Huddle, y en la misma se define la creación del listado *Sprint Backlog.* Los *Backlog* no son más que las tareas a realizarse, que son dirigidas específicamente a la implementación del video-juego. Con la finalidad de esta etapa, se concluye el proceso de desarrollo del video-juego.

3.1 Fase de desarrollo (Sprint Backlog)

Definido por la metodología Huddle, la fase de desarrollo se inicia con una reunión entre los integrantes del equipo de desarrollo (*Sprint Meeting*), donde se analiza el listado de *Feature Log* de la iteración (*Sprint*) donde se trabajará, y a partir de cada *Feature Log* definido en la misma, se desglosan las tareas necesarias (*Sprint Backlog*) desde el punto de vista de desarrollo y producción. El *Sprint Backlog* no es más que la: "*lista de tareas que va a realizar el equipo en una iteración, para construir un incremento.*" (11), por este motivo, una vez que se han definido cada uno de los *Backlog*, se pasa al proceso de producción, ya directamente enfocado a cada uno de los *Backlog*. El cumplimiento de cada uno de los *Backlog* pertenecientes a un *Feature*, define el grado de completitud de este *Feature* y a su vez del *Sprint* en general (12). Es así que se puede ver en la Imagen 24, el árbol de jerarquía que se forma a partir de los artefactos según la metodología Huddle.

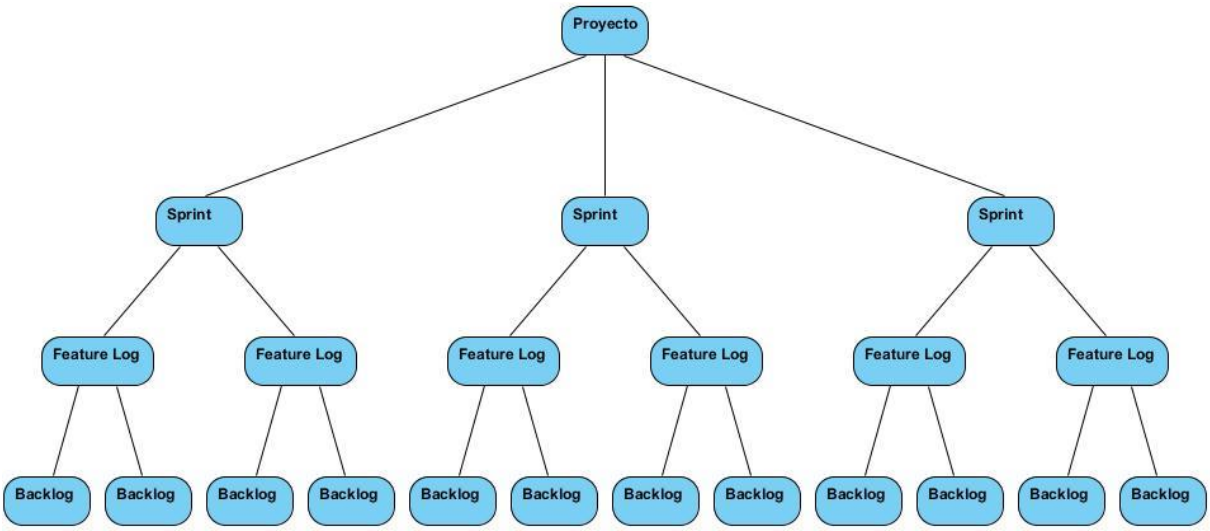


Imagen 24: Diagrama de jerarquía de los artefactos en Huddle

EL *Sprint Backlog* generado durante el proceso de desarrollo del presente trabajo, se puede encontrar en el Anexo 2.

3.2 Estilos de programación

3.2.1 Sangría y espaciado del código

Como buena práctica en la escritura de códigos entendibles y comprensibles por terceras personas, se utiliza la sangría y espaciado del código. Sangría se define como el corrimiento del código a la derecha (ya sea por espacios o tabulaciones) para reflejar la jerarquía del mismo. En el caso del espaciado del código, se basa en agregar saltos de línea al final de cada instrucción para hacer el código más legible. A continuación se muestra un ejemplo de las técnicas antes mencionadas en el proyecto actual:

```
public abstract class SceneBase extends Scene {
    public void loadTextures(){
        ScenesManager.getIntance().mActivity.getEngine().getTextureManager().loadTexture(mMap);
        for (BitmapTextureAtlas it:mFont)
            ScenesManager.getIntance().mActivity.getEngine().getTextureManager().loadTexture(it);
    }
    ...
}
```

Imagen 25: Ejemplo de sangría y espaciado

3.2.2 Documentación del código

En Java la documentación del código se realiza por medio de los identificadores:

- **//texto**: es utilizado para comentar una línea desde el lugar en que son puestos hasta el final de la misma.
- **/* texto */**: es utilizado para hacer bloques de comentarios capaces de tomar de 1 a más líneas.
- **/** @param texto */**: es utilizado para hacer comentarios destinados a la creación de la documentación Javadoc del código. Estos comentarios se colocan encima de la declaración de métodos o variables y para cada una de los posibles atributos (se definen con el símbolo @ delante del nombre).

A continuación se muestra un ejemplo donde se utiliza la documentación del código para apoyar la comprensión del mismo:

```
/**
 * Para activar/desactivar el sonido de la app
 */
public void sound_on_off(){
    /*
        Configuración de audio del video-juego
    */
    ScenesManager.getIntance().game_config.Sound = !ScenesManager.getIntance().game_config.Sound;
    ScenesManager.getIntance().game_config.update();
    //valores iniciales del audio
    if(ScenesManager.getIntance().game_config.Sound) {
        scene_music.setVolume(0.3f);
        for(Sound it:scene_sounds)it.setVolume(1f);
        sb_sound.setCurrentTileIndex(0);
    }else{
        scene_music.setVolume(0f);
        for(Sound it:scene_sounds)it.setVolume(0f);
        sb_sound.setCurrentTileIndex(1);
    }
}
```

Imagen 26: Ejemplo de comentarios en el texto

3.2.3 Nombre de variables y métodos

Es una de las prácticas de programación más distintivas en un código, puesto que el uso de nombres apropiados para las variables y los métodos utilizados, facilita la comprensión del código. Los casos en que los nombres utilizados son de la siguiente forma: ***a***, ***b*** o ***c***; pueden hacer imposible por parte de terceras personas la comprensión del código, mientras que el uso de nombres identificativos, pueden dar la medida de para lo que son destinadas las variables y permitir el desarrollo colaborativo entre varias personas. A continuación ejemplos de variables utilizadas en el proyecto.

```
public static final int MOVE_DOWN=456;//barra bajando
public static final int MOVE_UP=8755;//barra subiendo
public static final int DOWN=1238;//barra abajo
public static final int UP=567;//barra arriba
public static int EVENT_BAR=567;//estado actual de la barra superior
//operaciones con el tiempo del nivel
public static boolean HiloTime_STOP = false;//parar tiempo
public static boolean HiloTime_PAUSE = false;//pausar tiempo

//valores del nivel
public LevelManager LEVEL_MANAGER;
```

Imagen 27: Ejemplo de nombres de variables y métodos

3.2.4 Prefijos de control

Para llevar un control de los tipos de variables y la función que cumplen cada una en el código, se utilizan los prefijos de control. Esta es una técnica en la que a los nombres de las variables se les coloca un prefijo de identificación que puede variar en dependencia del tipo de variable y del objetivo de la misma. En el desarrollo del proyecto se utilizaron los prefijos de control que se encuentran en la siguiente tabla:

Control	Prefijo	Ejemplo en el código
Sprite	s	slogo
AnimatedSprite	sa	sapuntero
TiledSprite	st	sthand
TextureRegion	i	ilogo
TiledTextureRegion	it	itpuntero
Font	f	ftipografia
ChangeableText	tc	tctime
Text	t	tpoint
SpriteBackground	sb	sbfondo
BitmapTextureAtlas	m	mMap

Tabla 8: Prefijos de control

3.3 Pruebas unitarias

Tienen como objetivo probar de forma independiente los diferentes módulos de la aplicación. Definiéndose como módulo a un conjunto de funciones entrelazadas que cumplen con una meta (26). Para la realización de las pruebas, se hace uso de distintas aplicaciones que permiten automatizar el proceso e incluso la creación de pruebas aleatorias (27). Dentro de las herramientas que se pueden encontrar actualmente para realizar pruebas unitarias, se seleccionó JUnit, dado que es una de las herramientas más usadas en el lenguaje de programación Java y viene integrada por defecto en el IDE utilizado.

3.3.1 Resultados de las pruebas unitarias

Las pruebas realizadas al código de la aplicación fueron dirigidas a la clase de utilidades ***JUtilsFunctions***, cuyo principal objetivo es brindar funciones globales de uso común para la aplicación. A continuación se muestra un listado con los resultados de las pruebas:

Función	Descripción	Pruebas	Resultados
distance	Distancia entre dos puntos	100 pruebas aleatorias	0 errores
Mul	Multiplicar dos vectores	100 pruebas aleatorias	0 errores
Mod	Módulo de un vector	100 pruebas aleatorias	0 errores
angleVV	Ángulo entre dos vectores	100 pruebas aleatorias	0 errores

Tabla 9: Pruebas unitarias

3.4 Pruebas funcionales

Son realizadas con el objetivo de comprobar el correcto funcionamiento del software. Las mismas se aplican generalmente después de las pruebas unitarias, para comprobar la integración de los elementos probados. Las pruebas funcionales no comprueban la eficiencia o calidad del código, sino el correcto funcionamiento del mismo, de ahí que se les suela llamar: pruebas de caja negra (28).

Para la aplicación de las pruebas funcionales en el presente trabajo, se seleccionaron 4 dispositivos con SO Android (Tabla 10) de variadas especificaciones de memoria RAM, microprocesador y resolución. Al finalizar cada iteración en el proceso de desarrollo, se probaron las funcionalidades agregadas y la integración de las mismas con las creadas en iteraciones anteriores (28).

Modelo	Versión Android	Tamaño/Resolución	Memoria RAM	Microprocesador
Samsung GT-S5360T	2.3.6	3” / 240 x 320 px	256 MB	ARMv6 832MHz

Alcatel One Touch Pop C3 3330x	4.2.2	4” / 480 x 800 px	512 MB	ARM Cortex-A7 (1.3GHz)
BLU Life Play	4.2	4.7” / 720 x 1280 px	1.0 GB	Quad-core MediaTek 1.2GHz
Samsung Galaxy Note 4	5.0.1	5,7” / 2560 x 1440 px	1.0 GB	Quad-core 2.7 GHz

Tabla 10: Dispositivos utilizados en las pruebas de rendimiento

3.4.1 Resultados de las pruebas funcionales

Luego de aplicadas las pruebas funcionales al finalizar cada iteración del proceso de desarrollo, se pudieron encontrar los siguientes errores (Imagen 29):

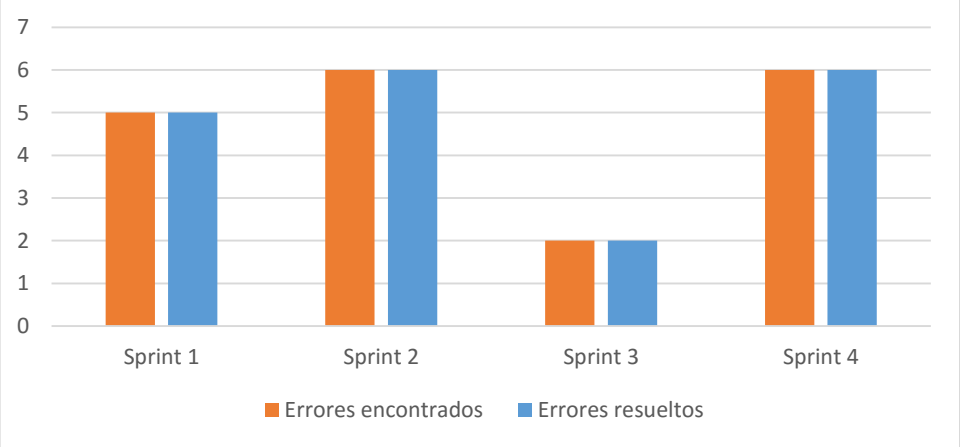


Imagen 28: Gráfica de errores encontrados en las pruebas funcionales

Cada error encontrado fue registrado en el listado de *Buglist* (como se especifica en la metodología Huddle) y se pueden encontrar en el Anexo 3.

3.5 Pruebas de rendimiento

Las pruebas de rendimiento tienen como objetivo hacer un chequeo del comportamiento de la aplicación cuando se ve sometida a una carga que actúe de forma concurrente sobre las funcionalidades de la misma (29). Aunque las pruebas funcionales chequean el rendimiento aislado del software, por medio de las pruebas de rendimiento se chequea la cohesión de las funcionalidades, además de otros aspectos como son el comportamiento del software ante el cambio de dispositivo o de SO (30).

3.5.1 Resultados de las pruebas de rendimiento

Luego de aplicadas las pruebas de rendimiento al video-juego “Recicla conmigo” en cada uno de los dispositivos seleccionados en la Tabla 10, teniendo en cuenta las características: uso del microprocesador (CPU), uso de memoria virtual (RAM) y cantidad de *frame* (cuadros) por segundo (FPS) de actualización de la pantalla, se arrojaron los siguientes resultados:

Dispositivo	Uso máximo de CPU	Uso máximo de RAM	Media de FPS
Samsung GT-S5360T	70 %	20 Mg	30 FPS
Alcatel One Touch Pop C3 3330x	23 %	20 Mg	32 FPS
BLU Life Play	20 %	20 Mg	40 FPS
Samsung Galaxy Note 4	10 %	20 Mg	60 FPS (velocidad ideal)

Tabla 11: Resultados de las pruebas de rendimiento

Con el término la prueba y como parte del proceso, se identificaron los siguientes errores:

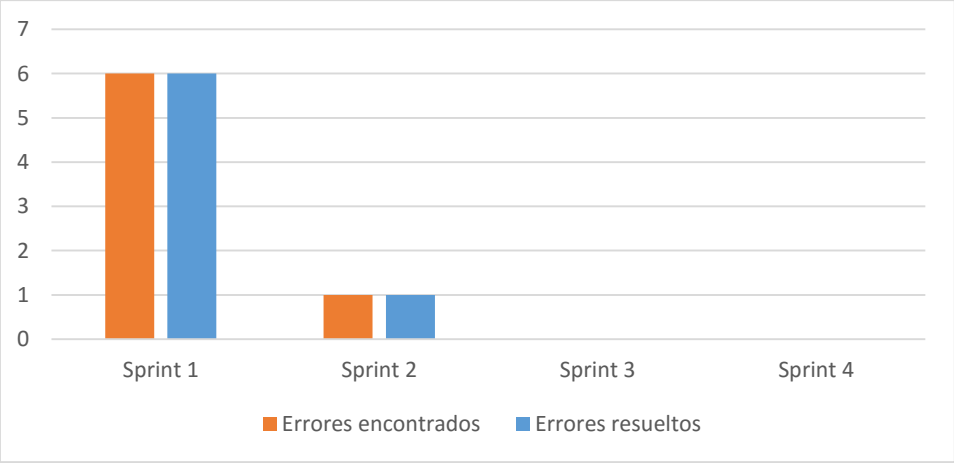


Imagen 29: Gráfico de errores encontrados en las pruebas de rendimiento

Cada error encontrado durante las pruebas de rendimiento, fue registrado en la tabla de *Buglist* que se encuentran en el Anexo 3.

3.6 Análisis Postmortem

Luego de concluida la fase de implementación y prueba de la aplicación, se pasa a la liberación del producto final al cliente. Una vez liberado el software y con vistas a mejorar la experiencia del equipo, se realiza el análisis Postmortem. Con el análisis Postmortem se hace un levantamiento de los aspectos positivos y negativos durante todo el proceso de desarrollo, para apoyar con la experiencia acumulada de dicho software a los venideros proyectos. A continuación se hace un resumen de los aspectos positivos y negativos encontrados en el desarrollo del video-juego “Recicla conmigo”:

Nombre	**Descripción**
Tamaño de los atlas de memoria en AndEngine.	Se pudo constatar, que el tamaño máximo que se puede definir para un atlas de memoria no puede exceder los 4096x4096 px, dado que se usaría demasiada memoria virtual durante la ejecución del video-juego y en dispositivos de pocas prestaciones las escenas se mostrarían en blanco.
Eliminación de elementos visuales en la escena.	Luego de encontrarse un error al intentar eliminar los desechos durante la ejecución del juego, se pudo determinar que no es posible eliminar los elementos visuales si no es de forma segura por medio del evento de actualización de los *frames* (cuadros), que se activa automáticamente por AndEngine. De no hacerse de esta forma, pueden existir colisiones entre el uso del elemento y la eliminación, deteniéndose la aplicación.
Medir el rendimiento del video-juego.	Para las pruebas de rendimiento se hizo un estudio de las herramientas que se podían utilizar, para concluirse que el IDE de desarrollo Android Studio, brinda todas las herramientas necesarias para hacer el chequeo del rendimiento del software.
Uso de la base de datos.	Con las pruebas de rendimiento se pudo constatar que al iniciar un nivel y extraer los datos de la BD, el tiempo de carga se hacía largo en dispositivos de bajas prestaciones, por este motivo se optó por cargar todos los niveles de la BD desde el inicio del video-juego, decisión que haría aumentar el uso de memoria virtual usada pero

	a su vez disminuiría el uso de CPU y los tiempos de cargado serían menores.

Tabla 12: Análisis Postmortem

Conclusiones parciales

Con el término del presente capítulo se llegó a las siguientes conclusiones parciales:

- Los *Sprint Backlog,* permiten llevar a cabo el proceso de desarrollo del video-juego de forma eficiente, organizando las tareas de implementación.
- Con las buenas prácticas de programación, es posible crear un software legible por terceras personas, en caso de futuras versiones de dicho video-juego.
- Las pruebas de software aplicadas, permitieron validar el video-juego constatándose la calidad del mismo.

CONCLUSIONES GENERALES

Con la culminación de esta investigación se arriba a las siguientes conclusiones:

- El estudio de los video-juegos relacionados con el reciclaje, permitió conocer las tendencias y diseños aplicados en esta área.
- Con el desarrollo del video-juego “Recicla conmigo” en SO Android, se logró aumentar el uso y disponibilidad del mismo.
- El análisis y diseño del video-juego a partir de la metodología Huddle, definió los pasos a seguir durante el proceso de desarrollo permitiendo el buen desenvolvimiento del mismo.
- Se aplicaron pruebas de calidad al software, pudiéndose constatar la calidad del video-juego desde el punto de vista funcional y de rendimiento.

RECOMENDACIONES

A partir del conocimiento adquirido durante todo el proceso de investigación, diseño y desarrollo que se llevó a cabo para implementar el video-juego “Recicla conmigo”, se pueden dar las siguientes recomendaciones, con vistas a versiones futuras del proyecto:

- Inclusión de un servicio web en la universidad que brinde actualizaciones del video-juego. Estas actualizaciones pueden ser desde nuevos niveles, hasta nuevos logros y puntuaciones.
- Llevar el video-juego “Recicla conmigo” a otros idiomas para que esté disponible a una mayor cantidad de usuarios.

GLOSARIO DE TÉRMINOS

Android: Sistema operativo orientado a dispositivos móviles, basado en una versión modificada del núcleo Linux. Inicialmente fue desarrollada por la empresa Android inc., que posteriormente fue comprada por Google. Actualmente lo desarrollan los miembros de la *Open Handset Alliance* (liderada por Google).

API (Application Programming Interface): Es un conjunto de subrutinas, funciones y procedimientos que brinda una biblioteca para ser utilizados por otro software como capa de abstracción.

Arcade: Término genérico para referirse a las máquinas recreativas. Género de juegos que por estética y/o sencillez de uso, recuerda a los de las máquinas recreativas.

Encapsulamiento: En la Programación Orientada a Objetos (POO) se le denomina encapsulamiento al ocultamiento de estados para que solo se pueda acceder a los atributos y métodos disponibles, aislando al objeto del exterior.

Game engine: Traducido del inglés significa motor de juego. Término usado en informática para referir las herramientas que permiten la creación de juegos, haciendo uso de funcionalidades y facilidades preestablecidas, que evitan la programación a bajo nivel.

Javadoc: Utilidad de Oracle para la generación de documentación APIs en formato HTML a partir del código fuente en Java. Es generado por la mayoría de los IDE actuales.

Multiplataforma: Aplicación o programa informático que puede utilizarse en diversos entornos o sistemas operativos.

NodeJS: Entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor basado en el lenguaje de programación ECMAScript, asíncrono, con entrada/salida de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google.

Oracle: *Oracle Corporation* es una compañía de software especializada en base de datos y sistemas de gestión de base de datos.

Osciloscopio: Herramienta utilizada para la medición de ondas, representándolas de forma gráfica en una pantalla.

Qt: Es una biblioteca multiplataforma, bajo licencia LGPL de código abierto y utiliza el lenguaje de programación C++ de forma nativa. Se utiliza generalmente en sistemas informáticos empotrados como automoción, aeronavegación y aparatos domésticos.

Ratio: Relación cuantificada entre dos magnitudes que representa la proporción entre ambas.

Renderizar: Término utilizado en la informática, para llamar al proceso de generar una imagen a partir del cómputo de diferentes procesos.

Video-consola: Sistema eléctrico para ejecutar video-juegos contenidos en algún dispositivo de almacenamiento, como memorias, discos, casetes, entre otros.

Video-juego: Juego electrónico, destinado a uno o más usuarios que se ejecuta desde una plataforma, que pueden ser las computadoras, los celulares o las video-consolas.

REFERENCIAS BIBLIOGRÁFICAS

1. Shaylor, N. Enigma and the Turing Bombe. Internet archive Wayback Machine. [En línea] 12 de 12 de 2012. [Citado el: 20 de 1 de 2016.] http://web.archive.org/web/20071212044158/http://frode.home.cern.ch/frode/crypto/Shaylor/bombe.html.

2. HISTORIA DE LAS COMPUTADORAS. García, Gerardo Ignacio Hernández. México : s.n., 2011.

3. Historia de la Computación. Monografias.com. [En línea] 20 de 8 de 2013. [Citado el: 25 de 1 de 2016.] http://www.monografias.com/trabajos/histocomp/histocomp.shtml.

4. Ray. Historia de los Videojuegos: El Origen y los Inicios. otakufreaks.com. [En línea] 10 de 10 de 2011. [Citado el: 1 de 2 de 2015.] http://www.otakufreaks.com/historia-de-los-videojuegos-el-origen-y-los-inicios/.

5. Historia de los Videojuegos: Empresas. http://indicelatino.com. [En línea] [Citado el: 5 de 2 de 2015.] http://indicelatino.com/juegos/historia/empresas/.

6. (FIB), Facultad de Informática de Barcelona. Historia de los videojuegos. Retro Informática. [En línea] 20 de 3 de 2008. [Citado el: 5 de 2 de 2015.]

7. Google. Google Play. [En línea] https://play.google.com.

8. Criterios de selección de metodologías de desarrollo de software. Tinoco Gómez, Oscar, Rosales López, Pedro Pablo y Salas Bacalla, Julio. Perú : s.n., 2010.

9. Letelier, Patricio y Penadés, Mª Carmen. Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP). www.cyta.com.ar/ta0502/v5n2a1.htm. [En línea] 15 de 4 de 2006. [Citado el: 10 de 2 de 2016.] http://www.cyta.com.ar/ta0502/v5n2a1.htm.

10. Bravo, Muñoz y Bolívar, Jimmy. Estudio de las fases de ciclo de desarrollo de software educativo con empleo de metodología ágil SCRUM. repositorio.utmachala.edu.ec. [En línea] 20 de 11 de 2015. [Citado el: 20 de 2 de 2016.] http://repositorio.utmachala.edu.ec/handle/48000/3572.

11. Scrummanager. Modelo original de Scrum para desarrollo de software - Scrum Manager BoK. Scrummanager. [En línea] 5 de 3 de 2013. [Citado el: 25 de 2 de 2016.] http://www.scrummanager.net/bok/index.php?title=Modelo_original_de_Scrum_para_desarrollo_de_software.

12. PROCESOS DE DESARROLLO PARA VIDEOJUEGOS. Morales Urrutia, Gerardo Abraham, y otros. Ciudad Juárez : CULCyT, 2015. 2007-0411.

13. The Java™ Tutorials. http://docs.oracle.com. [En línea] 2015. [Citado el: 27 de 2 de 2016.] http://docs.oracle.com/javase/tutorial/getStarted/TOC.html.

14. libgdx. libgdx.badlogicgames.com. [En línea] [Citado el: 5 de 3 de 2016.] https://libgdx.badlogicgames.com/.

15. AndEngine - Android Game Engine. www.andengine.org. [En línea] [Citado el: 5 de 3 de 2016.] http://www.andengine.org/blog/.

16. Unity - Game Engine. unity3d.com. [En línea] [Citado el: 5 de 3 de 2016.] https://unity3d.com/es.

17. Meet Android Studio. developer.android.com. [En línea] [Citado el: 10 de 3 de 2016.] https://developer.android.com/studio/intro/index.html.

18. SQLite Home Page. www.sqlite.org. [En línea] [Citado el: 10 de 3 de 2016.] https://www.sqlite.org/.

19. DB Browser for SQLite. sqlitebrowser.org. [En línea] [Citado el: 10 de 3 de 2016.] http://sqlitebrowser.org/.

20. Software Design Tools for Agile Teams, with UML, BPMN and More. www.visual-paradigm.com. [En línea] [Citado el: 11 de 3 de 2016.] https://www.visual-paradigm.com/.

21. GIMP. www.gimp.org. [En línea] [Citado el: 11 de 3 de 2016.] http://www.gimp.org/.

22. GIMP Descargas, tutoriales y foros. Alternativa a Photoshop gratis y libre. www.gimp.org.es. [En línea] [Citado el: 12 de 3 de 2016.] http://www.gimp.org.es/.

23. Inkscape. inkscape.org. [En línea] [Citado el: 15 de 3 de 2016.] https://inkscape.org/es/.

24. Pressman, Roger S. Ingeniería del software: Un enfoque práctico. s.l. : Mc Graw Hill, 2005. 0-07-285318-2.

25. Larman, Craig. UML y Patrones, Introducción al análisis y diseño orientado a objetos. México : Dawn Speth White, 1999. 970-17-0261-1.

26. Oré B., Ing. Alexander. Pruebas Unitarias. www.calidadysoftware.com. [En línea] [Citado el: 20 de 4 de 2016.] http://www.calidadysoftware.com/testing/pruebas_unitarias1.php.

27. Barrientos, Pablo Andrés. Enfoque para pruebas de unidad basado en la generación aleatoria de objetos. s.l. : sedici.unlp.edu.ar, 2014.

28. Pruebas de Caja Negra: Una Experiencia Real en Laboratorio. López, Carlos, Marticorena, Raul y Martín, David H. 2005.

29. Díaz, Francisco Javier, y otros. Usando Jmeter para pruebas de rendimiento. s.l. : XIV Congreso Argentino de Ciencias de la Computación, 2008.

30. Towards virtualized and automated software performance test architecture. Gwang-Hun, Kim, Yeon-Gyun, Kim y Kyung-Yong, Chung. 2013.

31. Metodologías ágiles en el desarrollo de aplicaciones para dispositivos móviles. Estado actual. Balaguera, Yohn Daniel Amaya. Bogotá D.C : revistas.unbosque.edu.co, 2016. 1692-1399.

32. Validación de Patrones de Diseño de Comportamiento a través de Perfiles UML: Observer, un caso de estudio. Cortez, Alberto Alejandro, y otros. 2014.

33. El patrón de arquitectura n-capas con orientación al dominio como solución en el diseño de aplicaciones empresariales. Cárdenas Escalante, Lain. 2013.

ANEXOS

Anexo 1: Feature Log

Feature ID	Sprint ID	Nombre	Días	Comentarios
F01	S01	Escena presentación (*splash*)	1	Escena inicial en que se muestra el progreso de cargado de los recursos del juego en la memoria.
F02	S01	Escena historia	1	Escena en que se muestra la historia del juego por medio de imágenes.
F03	S01	Escena cargar (*loading*)	1	Escena intermedia que se muestra mientras se carga en memoria el nivel que se va a jugar.
F04	S01	Escena del juego (*game*)	1	Escena donde se van a mostrar los niveles del juego.
F05	S01	Relación entre escenas	1	Crear las transiciones entre escenas y los botones de cambio de escena en el caso de ser necesitado.

Anexo 1.1: Feature Log del Sprint 1

Feature ID	Sprint ID	Nombre	Días	Comentarios
F06	S02	Utilización de imágenes animadas	1	Crear imágenes animadas y permitir controlarlas.
F07	S02	Cambio de imagen de fondo	1	En la escena historia, hacer los cambios de imágenes de fondo para mostrar el hilo de la historia.

F08	S02	Mover objeto por la escena	2	Crear objetos movibles por la escena, para ser aplicados como los recursos reciclables del juego.
F09	S02	Cargar recursos concurrentemente	1	En las escenas de cargado (*splash* y *loading*), crear los hilos consecuentes, para realizar las acciones de cargado concurrentemente.
F10	S02	Colisión entre objetos	1	Identificar colisión entre dos objetos, para ser aplicado en la acción de botar basura.

Anexo 1.2: Feature Log del Sprint 2

Feature ID	Sprint ID	Nombre	Días	Comentarios
F11	S03	Crear la BD del video-juego	3	Crear la BD del juego y definir los campos necesarios para almacenar lo indispensable.
F12	S03	Crear conexión entre el la BD y el juego	3	Crear los métodos y clase necesarios para la interconexión con la BD.

Anexo 1.3: Feature Log del Sprint 3

Feature ID	Sprint ID	Nombre	Días	Comentarios
F13	S04	Guardado y cargado del video-juego	2	Acción de cargar y guardar automáticamente el juego al cerrar mientras se está jugando.
F14	S04	Puntuaciones del juego	2	Mostrar la puntuación actual del nivel y el acumulado general al finalizar el mismo.

F15	S04	Sonido y audio	3	Agregar el audio y el sonido a la aplicación y las opciones de activar/desactivar.
F16	S04	Vibración	1	Agregar suplemento de vibración durante el juego.
F17	S04	Diseño del *splash* y del *loading*	2	Diseñar las escenas *splash* y *loading* para mejorar la estética.
F18	S04	Ayuda	1	Añadir ayuda al menú superior.
F19	S04	Reloj del juego	1	Añadir reloj al menú inferior.
F20	S04	Pistas del juego	1	Añadir lupa al menú inferior, para activar las pistas en el juego.

Anexo 1.4: Feature Log del Sprint 4

Anexo 2: Sprint Backlog

Backlog ID	Feature ID	Rol	Nombre de la tarea
B001	F01	Desarrollador	Crear clase de escena *splash*.
B002	F01	Desarrollador	Crear barra de progreso.
B003	F01	Diseñador	Añadir logotipo a la escena.
B004	F02	Desarrollador	Crear clase de escena historia.
B005	F02	Desarrollador	Agregar imágenes de la historia.
B006	F02	Desarrollador	Agregar botón siguiente.
B007	F02	Desarrollador	Agregar botón anterior.
B008	F02	Desarrollador	Agregar botón saltar historia.
B009	F03	Desarrollador	Crear clase de escena *loading*.

B010	F04	Desarrollador	Crear clase de escena *game*.
B011	F04	Diseñador	Agregar menú inferior
B012	F04	Diseñador	Agregar menú superior
B013	F05	Desarrollador	Crear enlace entre escena *splash* e historia.
B014	F05	Desarrollador	Crear enlace entre escena historia y juego.
B015	F06	Diseñador	Editar imágenes de los contenedores.
B016	F06	Desarrollador	Agregar contenedores a la escena *game* y configurar animación.
B017	F06	Diseñador	Editar objetos animados de los niveles.
B018	F06	Desarrollador	Agregar objetos animados de los niveles a la escena *game*.
B019	F07	Desarrollador	Programar botón siguiente de la escena historia.
B020	F07	Desarrollador	Programar botón anterior de la escena historia.
B021	F08	Desarrollador	Crear objeto *MoveSprite* con los atributos necesarios para el juego.
B022	F08	Desarrollador	Programar acción *drag&drop* del objeto *MoveSprite*.
B023	F09	Desarrollador	Crear *thread* del *splash* para cargar recursos del juego concurrentemente.
B024	F09	Desarrollador	Crear *thread* del *loading* para cargar nivel en la memoria del *engine*.
B025	F10	Desarrollador	Crear clase ***JUtilsFunction*** para funciones globales de la aplicación.

B026	F10	Desarrollador	Agregar método de colisión entre objetos a la clase ***JUtilsFunction*.**
B027	F11	Desarrollador	Crear tablas a partir del modelo de datos definido en el diseño.
B028	F11	Desarrollador	Agregar primer nivel a la BD, para pruebas en la aplicación.
B029	F12	Desarrollador	Crear clase ***MainDatabase*** para la conexión de la aplicación a la BD.
B030	F12	Desarrollador	Agregar a la clase ***MainDatabase*,** soporte para utilizar la BD del juego.
B031	F12	Desarrollador	Crear clases definidas en el diseño para el trabajo con los datos de la BD.
B032	F12	Desarrollador	Cargar niveles de la BD en la escena *splash*.
B033	F13	Desarrollador	Crear tabla **config** en al BD para las configuraciones del juego.
B034	F13	Desarrollador	Crear clase ***DBConfig*** de enlace con la tabla ***config***.
B035	F13	Desarrollador	Crear métodos carga/guardar en la clase ***DBConfig***.
B036	F14	Desarrollador	Agregar puntuaciones a la clase *game*.
B037	F14	Desarrollador	Agregar puntuaciones a la clase ***ALevel***.
B038	F14	Desarrollador	Mostrar puntuaciones en el menú inferior.
B039	F15	Desarrollador	Agregar sonido a los recursos.
B040	F15	Desarrollador	Implementar sonido en la escena *game* e historia.

B041	F15	Desarrollador	Guardar estado del sonido por medio de la clase ***DBConfig*** para las configuraciones.
B042	F16	Desarrollador	Agregar vibración a efectos secundarios del juego.
B043	F17	Diseñador	Diseñar barra de progreso de la escena splash.
B044	F17	Diseñador	Diseñar animación para la escena *loading*.
B045	F18	Desarrollador	Agregar ayuda al menú superior.
B046	F18	Diseñador	Crear mensaje de ayuda.
B047	F19	Desarrollador	Agregar *thread* para el tiempo del juego.
B048	F19	Desarrollador	Crear atributos globales para pausar o detener el *thread* del tiempo.
B049	F20	Desarrollador	Añadir botón lupa a la barra inferior.
B050	F20	Desarrollador	Programar elección aleatoria de elemento para ser señalado al usar las pistas.
B051	F19	Desarrollador	Arreglar reloj (poner contador inverso).
B052	F04	Desarrollador	Arreglar evento GAME_LOSER de la clase nivel.
B053	F04	Diseñador	Resumen final del juego.
B054	F04	Diseñador	Arreglar último nivel (problemas de diseño).
B055	F15	Desarrollador	Arreglar botón de activar/desactivar sonido.
B056	F20	Desarrollador	Arreglar pistas del juego (problemas con la elección aleatoria del recurso).

B057	F04	Desarrollador	Poner recurso por encima de todo al ser arrastrados al contenedor.
B058	F04	Diseñador	Arreglar clase **Mensajes**, del juego.
B059	F04	Diseñador	Crear ícono del juego.
B060	F10	Desarrollador	Aumentar área de acción de los *sprite* en el juego.
B061	F04	Desarrollador	Mensaje de confirmación al cerrar el juego.
B062	F02	Desarrollador	Agregar *swipe* a la escena historia.

Anexo 2.1: Listado de Backlog

Anexo 3: Buglist

Bug ID	Feature ID	Descripción	Descripción técnica	Autor
BL001	F09	Al cargar segundo nivel, se queda en blanco la pantalla.	Problemas con la memoria al cargar objetos de más de 2 MB.	Guillermo
BL002	F09	Problema de profundidad con los objetos del nivel.	Falta el campo Z para la profundidad de los objetos.	Guillermo
BL003	F07	Problema de la barra superior, no sale a partir del segundo nivel.	No se actualizó la profundidad de los objetos.	Guillermo
BL004	F11	Objetos superpuestos en la escena.	Problemas con la ubicación de los objetos en el atlas de memoria.	Guillermo
BL005	F04	Error con los botones de los mensajes	Mal orden al deshabilitar la escena al cerrar los mensajes.	Guillermo

BL006	F04	Nivel de entrenamiento, error al pasar segundo mensaje.	Falta desactivar el hand.	Guillermo
BL007	F04	Nivel de entrenamiento, error al pasar tercer mensaje.	Falta desactivar el hand.	Guillermo
BL008	F19	Error con el tiempo al pasar nivel. Se queda andando el del nivel anterior.	Error al terminar el thread del tiempo.	Guillermo
BL009	F10	Problemas al soltar los desechos en sus contenedores.	Problemas al eliminar los Sprite de la escena.	Guillermo
BL010	F04	Botón cerrar, problemas para activarlo.	Aumentar área de acción del botón.	Guillermo
BL011	F04	Botón ayuda, problemas para activarlo.	Aumentar área de acción del botón.	Guillermo
BL012	F04	Botón audio, problemas para activarlo.	Aumentar área de acción del botón.	Guillermo
BL013	F04	Botón lupa, problemas para activarlo.	Aumentar área de acción del botón.	Guillermo
BL014	F04	Botón hand, problemas para activarlo.	Aumentar área de acción del botón.	Guillermo
BL015	F08	Recursos no se ponen por encima al ser arrastrados.	Problemas con la profundidad de los objetos.	Guillermo
BL016	F20	Lupa no funciona algunas veces.	Error en la función aleatoria de selección de los recursos.	Guillermo

BL017	F15	Audio no funciona al pasar de nivel.	Problemas con la pausa del audio.	Guillermo
BL018	F14	Puntuación no muestra más de un dígito.	Problema con el ChangeableText de la puntuación.	Guillermo
BL019	F08	Recursos pequeños difíciles de seleccionar.	Aumentar radio de acción de los recursos.	Guillermo
BL020	F04	Error ortográfico en mensaje de inicio.		Guillermo
BL021	F05	Solo es posible pasar el juego una vez, en la segunda da error.	Problemas con la liberación de la memoria usada en los niveles.	Guillermo
BL022	F04	Último nivel se queda en blanco al terminarlo.	Problemas al liberar los recursos del nivel.	Guillermo
BL023	F13	Problemas en el mensaje de confirmación de cerrado.	Error en la función de guardar el nivel.	Guillermo
BL024	F14	Puntuación general no funciona.	Problemas al sumar la puntuación alcanzada en el nivel terminado.	Guillermo
BL025	F11	Problemas con las texturas de los niveles.	Arreglar valores de los niveles en la base de datos.	Guillermo

Anexo 5: listado de Buglist